REPORT DOCUMEI

Form Approved
OBM No. 0704-0188

Public reporting burden for this collection of information is estimated to av... ...ching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of i... ...lection of information. Including suggestions for reducing this burden, to Washington Headquarters Services, Directorat... ...te 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (070-

1

1. Agency Use Only *(Leave blank).*	**2. Report Di** 1992	...ed. rinal - Proceedings

4. Title and Subtitle. Comparison of Texture Analysis Techniques in Both Frequency and Spatial Domains for Cloud Feature Extraction	**5. Funding Numbers.** *Contract* *Program Element No.* 0602435N *Project No.* RM35G81 *Task No.* *Accession No.* DN656756 *Work Unit No.* 14312B
6. Author(s). Nahid Khazenie and Kim Richardson	
7. Performing Organization Name(s) and Address(es). Naval Research Laboratory Atmospheric Directorate Monterey, CA 93943	**8. Performing Organization Report Number.** PR 92:087:431
9. Sponsoring/Monitoring Agency Name(s) and Address(es). Office of Naval Technology Arlington, VA 22217	**10. Sponsoring/Monitoring Agency Report Number.** PR 92:087:431

11. Supplementary Notes.
Published Int. Society for Photogrammetry and Remote Sensing.

12a. Distribution/Availability Statement. Approved for public release; distribution is unlimited.	**12b. Distribution Code.**

13. Abstract *(Maximum 200 words).*

Identification of cloud through cloud classification using satellite observations is yet to produce consistent and dependable results. Cloud types are too varied in their geophysical parameters, as measured by satellite remote sensing instruments, to provide for a direct accurate classification. To aid in classification, texture measures are additionally employed. These measures characterize local spectral variations in images. They are widely used for image segmentation, classification, and edge detection. Numerous methods have been developed to extract textural features from an image on the basis of spatial and spectral properties of the image. In our effort, several of these methods are studied for their applicability in cloud classification and cloud feature identification. These examined texture methods include a) spatial gray-level co-occurrence matrices, b) gray-level difference vector method, and c) a class of filters known as Gabor transforms. Methods)a and b) are spatial and statistical while method c) is in the frequency domain. A series of comparative tests have been performed applying these methods to NOAA-AVHRR satellite data. A discussion as to the suitability of these texture methods for clout classification concludes this study.

14. Subject Terms. Objective analysis, initialization, satellite data	**15. Number of Pages.** 6
	16. Price Code.

17. Security Classification of Report. Unclassified	**18. Security Classification of This Page.** Unclassified	**19. Security Classification of Abstract.** Unclassified	**20. Limitation of Abstract.** SAR

ISPRS
WASHINGTON, D.C.
1992

International Society for Photogrammetry and Remote Sensing XVIIth Congress
Société Internationale de Photogrammétrie et de Télédétection XVII^e Congrès
Internationale Gesellschaft für Photogrammetrie und Fernerkundung XVII. Kongress

ISSN 0256-1840

INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY AND REMOTE SENSING

ARCHIVES INTERNATIONALES DE PHOTOGRAMMETRIE ET DE TELEDETECTION

INTERNATIONALES ARCHIV FÜR PHOTOGRAMMETRIE UND FERNERKUNDUNG

VOLUME
VOLUME
BAND
XXIX

PART
TOME
TEIL
B7

COMMISSION
COMMISSION
KOMMISSION
VII

ISPRS

Edited by: Lawrence W. Fritz, Congress Director
James R. Lucas, Technical Program Chairman

Published by the Committee of the XVII International Congress for Photogrammetry and Remote Sensing
Publié par le Comité du XVII^e Congrès Internationale de Photogrammétrie et de Télédétection
Herausgegeben vom Komitee für den XVII. Internationalen Kongress für Photogrammetrie und Fernerkundung

COMPARISON OF TEXTURE ANALYSIS TECHNIQUES IN BOTH FREQUENCY AND SPATIAL DOMAINS FOR CLOUD FEATURE EXTRACTION

Nahid Khazenie[1,2]

[1]University Corporation for Atmospheric Research,
Boulder, CO 80301

Kim Richardson[2]

[2]Naval Research Laboratory, Monterey, CA 93943-5006

ABSTRACT

Identification of cloud types through cloud classification using satellite observations is yet to produce consistent and dependable results. Cloud types are too varied in their geophysical parameters, as measured by satellite remote sensing instruments, to provide for a direct accurate classification. To aid in classification, texture measures are additionally employed. These measures characterize local spectral variations in images. They are widely used for image segmentation, classification, and edge detection. Numerous methods have been developed to extract textural features from an image on the basis of spatial and spectral properties of the image. In our effort, several of these methods are studied for their applicability in cloud classification and cloud feature identification. The examined texture methods include a) spatial gray-level co-occurrence matrices, b) gray-level difference vector method, and c) a class of filters known as Gabor transforms. Methods a) and b) are spatial and statistical while method c) is in the frequency domain. A series of comparative tests have been performed applying these methods to NOAA-AVHRR satellite data. A discussion as to the suitability of these texture methods for cloud classification concludes this study.

Key Words: texture analysis, cloud classification, Gabor transforms, spatial gray-level co-occurrence matrices, gray-level difference vector (GLDV), NOAA-AVHRR.

INTRODUCTION

Identification of cloud types by automated cloud classifiers, which operate on a pixel by pixel basis, has yet to show dependable and accurate results. Clouds have geophysical parameters which are too inconsistent, as measured by satellite remote sensing instruments, to provide for a direct accurate classification. No method developed to date provides a reliable spectral signature which would uniquely identify a specific cloud type anywhere on the Earth globe during any season. Cloud types vary in their spectral response at different latitudinal locations and at different times of the year. These variations complicate methods required for cloud type identification using remote sensing techniques.

Surveying the various available statistical, structural, and frequency domain techniques applied to cloud classification, it appears that there are not enough parametrization vectors to uniquely separate any one cloud type. For this reason, texture analysis methods are drawn upon in addition to aid in this problem. The use of texture parameters has been reported on extensively in recent literature (Wechsler, 1980). Texture techniques used in our study include a) spatial gray-level co-occurrence matrices, b) gray-level difference vector (GLDV) method, and c) a class of filters known as Gabor transforms. Each of these approaches has unique merit for providing additional information about cloud masses within a scene. These unique differences are the focus in this study.

Images in this case study are composites of Advanced Very High Resolution Radiometer (AVHRR) channel one and channel four. Pixel by pixel classifications of cloud types, based on the spectral and spatial responses from these channels, are enhanced with results from the various texture analysis algorithms. Results of the classifications from the combined techniques are compared and discussed.

DATA

An image from the Gulf of Alaska was chosen for this work. This region was selected due to its high latitude which presents challenging solar zenith angles. It also provides snow within the scene which tests snow and cloud separation capabilities of the candidate methods. Furthermore, the general meteorological activity within this region is high thereby presenting a continuous varying source of frontal cloud masses.

The scene selected for presentation is one of eight images used in this study. It is an AVHRR image from 15 October 1988, 19Z. A full resolution (1.1 km per pixel) sector of 1024 by 1024 ten-bit pixels was extracted from the original 2048 by 2048 data set.

The channel one and channel four radiance images are shown in Figures 1 and 2. The channel one image is histogram-equalized for purposes of display. The channel four image is inverted so as to represent clouds in lighter gray shades.

The large band of clouds in the extreme right of the image is a frontal cloud mass that has previously moved through the area. This cloud mass is characterized by high thick cirrus over cumulus. These clouds are brightened by their height as well as by the low sun angle which is characteristic for this northern latitude.

In the lower central portion of the image are well defined cloud streets. They are trailed by open cell stratocumulus and altostratus that extend to the left center of the image. The mixed layered cloud mass in the lower left portion of the image represents stratus and altostratus with a cover of thick cirrus. Some closed cell stratocumulus are at the bottom of the image between the stratus and frontal clouds.

Snow can be seen in the upper central portion of the image. Typically, snow will be observed to have a dendritic-like structure which distinguishes it from cloud masses.

TEXTURAL METHODS

Texture is a term used to characterize the surface of a given object. It can also be applied to an image of a phenomenon. It is undoubtedly one of the main features drawn upon in image processing and pattern recognition. Texture analysis plays a fundamental role in classifying objects and outlining significant regions of a given gray level image (Wechsler, 1980). Despite its ubiquity in image data, though, texture lacks a precise definition. Some definitions characterize texture as visual images which possess some stochastic structure. Other definitions describe texture as an attribute generated by a local periodic pattern. Whatever the definition, most algorithms which derive texture from an image fall into the categories of either statistical or frequency domain. A brief description of the three texture methods of interest follows.

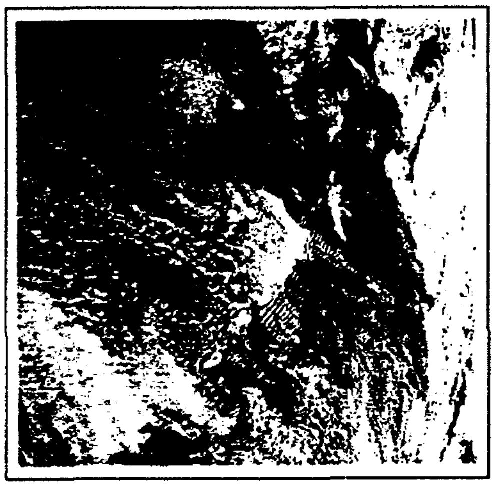


Figure 1. AVHRR channel 1 ten-bit radiance values, histogram equalized for display.

Figure 2. AVHRR channel 4 ten-bit radiance values, inverted for display.

Statistical Methods

The two most commonly used statistical texture methods are the a) gray-level difference vector (GLDV) method (Welch et al., 1990, Khazenie and Richardson, 1991), and the b) co-occurrence matrix method (Haralick, 1973). Our current study draws upon both of these methods. Both methods extract a set of statistical parameters from a given image. Some of the commonly extracted texture parameters are inertia, correlation, homogeneity, entropy, energy, variance, skewness, and kurtosis. These parameters are then used as the input features to a classifier.

Texture measures are derived commonly from statistical parameters of first or second order. The GLDV method estimates the probability density function for differences taken between image function values at locations spaced **d** pixels apart and at an angle θ. The resulting texture measures are based on this first order statistic. The spatial co-occurrence matrix method, on the other hand, estimates the joint gray level distribution for two gray levels located at a distance **d** and at an angle θ. The texture measures derived by the co-occurrence matrix method are based on this second order statistic.

The co-occurrence matrix method is used in this study to derive texture values of entropy, homogeneity, energy (similar to the GLDV angular second moment), and correlation. These four parameters were calculated for the radiances of each of the two channels, AVHRR channel one and channel four, for a total of eight texture values. Each texture value was processed using three different convolution sizes. The **n** by **n** convolution sizes are **n** = 3, 9, and 16. In addition, the search angle for each of the convolutions was varied to determine whether or not the derived textures possess any angular dependence. The angle was set to θ = 0,° 45,° 90,° and 135.° Search angle dependence is expected only when the surface resolution is much smaller than the 1.1 km surface resolution of the AVHRR instrument and indeed, as discussed later, no angular dependence was identified using the co-occurrence matrix method and the given image data.

The GLDV technique was similarly applied. The same texture values as for the co-occurrence matrix method were calculated. The calculations were performed on the same channel one and channel four radiance values, but only for a single search angle, θ = 0. The search angle non-dependence had already been established from working with the co-occurrence matrix method. Seven convolution sizes were chosen to derive the texture values of entropy, local homogeneity, and angular second moment. The convolution sizes were **n** = 3, 5, 9, 11, 16, 32, and 64. From a previous study (Khazenie and Richardson, 1991) the three sizes of **n** = 3, 16, and 64 provided the best statistical representation of the data for use in cloud classification. This finding was re-established in the current work.

Frequency Domain Methods

Spatial granularity and repetitiveness is one of the characteristic aspects of texture. Both can be quantified by looking at the frequency content of an image. It is therefore reasonable to expect that transform techniques are suitable for extracting texture information from images.

The Fourier transform analysis method (Lendaris et al., 1970) is a procedure which works in the frequency domain. It is, by far, the most used transform method. Image features, such as spectral rings or edges, are derived from the image power spectrum by this technique.

Related to the Fourier transform are functions first introduced by Gabor (Gabor, 1946). These functions have been extended to two dimensions (Daugman, 1980) resulting in what is known as the two-dimensional (2-D) Gabor filters.

One of the unique properties of Gabor filters is their ability to discriminate textural features in a way similar to that of human vision (Fogel et al., 1989). Another important property is their achievement of the theoretical lower bound of joint uncertainty in the two dimensions of visual space and spatial frequency variables (Bovik et al., 1990). Additional advantages of Gabor transforms include their tunable spatial orientation, radial frequency bandwidths, and tunable center frequencies.

The 2-D Gabor filter is a harmonic oscillator, a sinusoidal plane wave within a Gaussian envelope. The convolution version of the complex 2-D Gabor function has the following general form.

$$G(x,\ y \mid W,\ \theta,\ \varphi,\ X,\ Y) =$$

$$\left(\frac{1}{2\pi\sigma^2}\right) \cdot exp\left[\frac{-[(x-X)^2\ +\ (y-Y)^2]}{2\sigma^2}\right] \cdot$$

$$sin(W(xcos\theta\ -\ ysin\theta)+\varphi) \qquad (1)$$

In equation (1), the Gaussian width is σ, the filter orientation is θ, the frequency is W, and the phase shift is φ. Variables X and Y define the center of the filter.

The Gabor function, equation (1), can be represented as a complex function having a real and an imaginary component, G_1 and G_2, respectively.

$$G_1(x,\ y \mid W,\ \theta,\ \varphi\ =\ 0,\ X,\ Y)$$

$$G_2(x,\ y \mid W,\ \theta,\ \varphi\ =\frac{\pi}{2},\ X,\ Y)$$

Functions G_1 and G_2 are, respectively, even and odd symmetric along the preferred orientation direction θ. The results of convoluting G_1 and G_2 with any two-dimensional function are identical except for a spectral shift of $\pi/2$ along the direction θ.

Given an image I(x, y), its Gabor transformation for a given filter size n with orientation angle θ and frequency W is given by the following equation.

$$S^2(X,Y \mid W,\theta)\ =\ [G_1\ *\ I(x,y)]^2\ +\ [G_2\ *\ I(x,y)]^2$$

The Gabor filter described by equation (1) was applied to the AVHRR test images' channel one and channel four radiances. The response was evaluated for filters with $\theta = 0.°$ 45,° 90,° and 135.° The frequency W was set to $2\pi f/(n/2)$ where f = 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. The tested filter sizes n were 9, 17, 33, and 65. The best results for cloud typing from the four convolutions was for n = 17.

RESULTS

The 1024 by 1024 ten-bit radiance data from channels one and four was used as input to the various texture algorithms. The texture output was then resized back to the full 1024 by 1024 resolution and added, as supplementary channels, to the radiance data. The resulting N channel data set was classified using a standard statistical unsupervised classifier.

Co-occurrence Matrix

The texture results from the co-occurrence matrix algorithm were first classified alone for each of the convolution sizes and search angles. Figure 3 is the result of this classification for an n by n convolution size where n = 3 and for a search angle θ = 45.° This represents the best result for all of the classifications from the co-occurrence matrix algorithm.

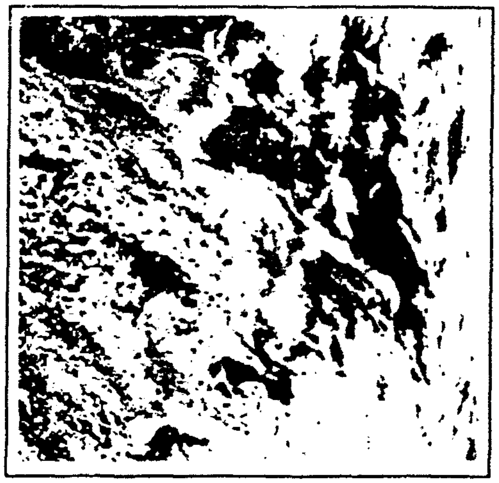


Figure 3. Texture values from the co-occurrence matrix algorithm, classified and scaled up for display.

It is clear from Figure 3 that the texture values alone do not *represent cloud types with any accuracy. Figure 3 shows nine* classes, but none identify any of the cloud types uniquely. The classified images are extremely noisy and at best represent features within the cloud masses rather than the cloud types themselves. There is, however, one reasonably accurate feature which resulted from this classification. For the lowest convolution size, at all search angles, the clear vs. cloudy areas are quite distinct.

For convolution sizes of n = 9 and 16, for all search angles, the classification separates the clear areas from the cloudy ones with success as well. It also produces a smoother classification. The cloud types, though, are still difficult to identify.

The classifications were performed twice on the texture results where n = 9 and 16. The first classification was performed on all of the texture values derived. The second classification was performed on the same values except for the correlation parameter. The results of the cloud type classification neither improved nor degraded. Therefore it seems that the correlation parameter does not contribute to the information needed for cloud typing.

In all cases of classifying only the co-occurrence matrix texture results, the thick cirrus was identifiable as a homogeneous feature, yet it was assigned the same class as portions of the open cell stratocumulus. Also in all cases, the snow was not separated from the clouds. The snow was assigned the same class as the stratus and altostratus clouds.

It was concluded at this point that classifying texture values alone does not provide sufficient results for identifying cloud types. The next step then was to provide more information to the classifier. The eight texture values were combined with the two AVHRR channel radiances (channel one and four) and classification was performed on the resulting ten channels of data.

The texture values for convolution size of $n = 9$ were resized to the full **1024** by **1024** resolution, equal to that of the channel radiances, merged with the channel radiances, and the resulting data set was classified. The output is shown in Figure 4.

The search angle was varied as before, but the results from the classifier showed virtually no differences for unequal angles. The results for varied search ^ngles were compared by calculating difference images. Only minor variations were noted in some of the mixed layer cloud types amounting for less than **1%** difference over the entire image. From this it was concluded that the process is not dependent on search angle and all further comparisons were made setting $\theta = 0.^{\circ}$

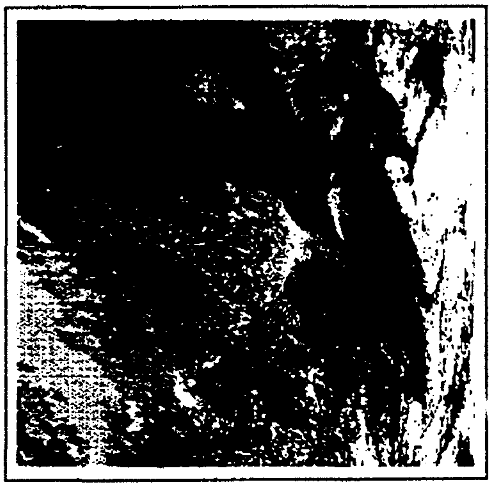


Figure 4. Combination of AVHRR channel 1, channel 4, and texture values from the co-occurrence matrix algorithm, classified.

The frontal cloud mass at the extreme right of the image in Figure 4 is represented by four distinct classes. The thick cirrus, the altocumulus, the cumulus, and the lower level stratocumulus each appear as distinct cloud types. They are affected by the sun angle thereby giving the cirrus over the frontal cloud mass a different class than the cirrus over the stratus in the lower left portion of the image.

The classes representing the stratus clouds provide more separation of cloud types than a human photointerperter would give. Should the goal be to duplicate human performance, one can easily combine some of the statistical classes. However, our goal was to obtain parameter vectors for performing unsupervised cloud classifications, no matter how many vectors there may be, as long as the distinct cloud types can be separated from each another. That goal was achieved.

Gray Level Difference Vector

The texture results from the gray level difference vector (GLDV) algorithm were first classified alone, identically as for the co-occurrence matrix. Similarly, the classification results from these texture values alone do not provide cloud type information directly. The results are essentially identical to those shown in Figure 3 for the co-occurrence matrix. The textures values, when classified, identify edges between features within the image well, but the features are various areas within the cloud type rather than the cloud type itself.

As with the co-occurrence matrix method, the GLDV performs very well at identifying cloud versus no cloud areas within the scene. It is not know at this time, however, if this capability can be extended easily to all AVHRR images. With more study this may indeed prove to be the case. Simple thresholding of the texture values may be all that is required. Results from a previous study (Khazenie and Richardson, 1991) support this conjecture.

As with the co-occurrence matrix output, the textures derived by the GLDV were then combined with the channel one and channel four radiances. The resulting eight channels were classified and the outcome is shown in Figure 5. Again, the results are essentially identical to those from the co-occurrence matrix method (Figure 4) in their ability to type clouds. Sun angle remains a problem within the frontal cloud region at the extreme right of the image. However, the mixed layer clouds in the lower left show the same successful level of cloud type separation as with the co-occurrence matrix method.

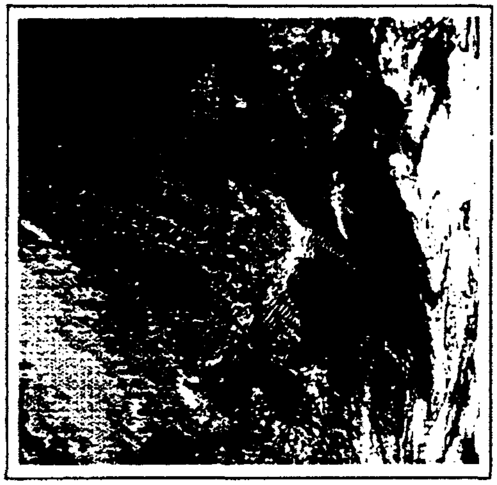


Figure 5. Combination of AVHRR channel 1, channel 4, and texture values from the GLDV algorithm, classified.

Convolution size plays a role in the ability to type clouds. For $n > 16$ the algorithm is able to identify the presence of clouds. It is also able to determine that the texture in the region is unique. However, it does not provide enough information to the classifier to separate cloud types. Although the statistical significance is in favor of the higher convolution sizes, it is the lower convolution sizes that provide the textural significance to the classifier for cloud type identification.

Gabor Filters

Figure 6 presents the result of classifying the test image channel one and channel four radiances combined with the Gabor filter output where $n = 17$ and phase angle $\varphi = 0$. Of the available **2048** by **2048** data, the same **1024** by **1024** scene was originally acquired as for the statistical methods. However, computer resources available for the study of Gabor filters could digest no more than **512** by **512** images. Therefore, only the lower left quarter of each **1024** by **1024** scene was analyzed. One such quarter is shown in Figure 6.

The thick cirrus over the stratus is well separated. This is a great improvement over the classification of texture values from the Gabor filter alone. Indeed the classifications of the combined image, radiances and textures shown in Figure 6, are much easier to label than are either of the classifications based on texture only.

Convolution sizes $n > 17$ do not perform well for a wide variety of cloud types within a scene. This follows along with the same findings as for the statistical textural methods. Important textural attributes in the cloud mass are lost when higher convolution sizes are used. With such convolution sizes, the textural analysis shows the dominant texture over each cloud mass, but does not sufficiently indicate characteristic features of that cloud mass thereby missing the classification of the cloud type. In particular, the open cell stratocumulus and altostratus within the image did not separate out at the higher convolution sizes.

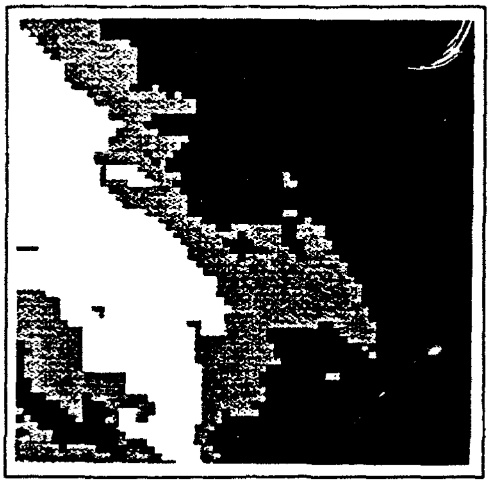


Figure 6. Combination of AVHRR channel 1, channel 4, and texture values from the Gabor filter, classified. The results shown cover only the lower left quarter of the test image.

While it has not been tested in this study directly, it is conjectured that the approach using the Gabor filter will be less sensitive to latitude variances and seasonal changes compared to the two statistical techniques. The Gabor filter is a tunable algorithm. With proper adjustment of control parameters it should be possible to desensitize the filter to local effects, such as latitude changes and seasonal effects, while retaining the ability to extract the required physical response which uniquely represents each cloud type.

Inter-comparisons

The results from classifying only the output of the two statistical textural methods, the co-occurrence matrix and the GLDV, are almost exactly alike. This is reasonable since the texture measures calculated by both of these algorithms were the same. One should expect the same results even though they were arrived at by different means. The GLDV is based on first order statistics while the co-occurrence matrix method is based on second order statistics. It can therefore be concluded that, given our test images, the extra complexity of the second order statistics is not necessary for arriving at satisfactory results.

It is also noteworthy that the results from the two statistical methods are virtually identical even though different convolution sizes were used. This suggests that the features, which distinguish cloud types, are fairly coarse. It also suggests a lack of fine features which would distract a method that uses a small convolution size.

The output of the Gabor filter has different characteristics compared to the output of the co-occurrence matrix and GLDV, yet the ability to separate cloud types is very similar for all three methods. The resolution of the Gabor filter output is lower, seventeen pixels versus nine for the co-occurrence matrix and three for the GLDV. The classification results are greatly influenced by this resolution difference. This is obvious by comparing the pixel size in Figure 6 with Figure 4. In Figure 6 the boundary of the thick cirrus over the stratus has been smoothed. This is also true for the stratus cloud types. The separation of multilayered clouds is similar for all three methods.

The computer processing time required for the Gabor filter method proved much less than either of the two statistical methods (co-occurrence matrix or GLDV). Processing a full **1024** by **1024** scene using the Gabor filter took approximately one minute on a SUN SparcStation II. The statistical methods required approximately ten minutes each for the same image.

CONCLUSION

Classification of cloud types using spectral and derived textural parameter vectors alone has not been completely successful. Additional information about texture in the image provides more input to a cloud classifier. Such an addition shows considerable improvement over cloud classification based only on spectral information. Despite the marked improvement, however, it does not yet appear that the addition of texture information provides all of the necessary parameters required to successfully classify and completely label cloud types. Nevertheless, results from this study indicate that Gabor filters applied to the spectral data set, and used in conjunction with the spectral data for classification, extract cloud types better and faster than the other techniques explored.

Acknowledgements

The authors extend their extreme appreciation to Mr. Mike Linda for his invaluable help and fortitude.

The NRL Contribution number is 92:087:431. The authors gratefully acknowledge the support of the sponsor, the Office of Naval Technology, Code 22, Mr. James Cauffman, Program Element 62435N. This paper is approved for public release; distribution is unlimited.

REFERENCES

Bovik, A. C., M. Clark, and W. S. Geisler, 1990. Multichannel Texture Analysis Using Localized Spatial Filters. *IEEE Trans. on Pattern Anal. Machine Intell.*, 12(1):55-73.

Daugman, J. D., 1980. Two Dimensional Spectral Analysis of Cortical Receptive Field Profiles. *Vision Res.*, 20:847-856.

Fogel, I., and D. Sagi, 1989. Gabor Filters as Texture Discriminators. *Biological Cybernetics*, 61(2):79-162.

Gabor, D., 1946. Theory of Communication. *J IEE*, 93:429-457.

Haralick, R. M., K. Shanmugam, and I. Dinstein, 1973. Textural Features for Image Classification. *IEEE Trans. Systems, Man and Cybernetics*, SMC-3(6):610-621.

Lamei, N., N. Khazenie, and M. M. Crawford, 1992. Multi-Spectral Texture Analysis for Cloud Feature Discrimination. *Proc. IGARSS'92 Symp.*, May 1992, Clear Lake, Texas.

Lendaris, G. O., and C. L. Stanley, 1970. Diffraction Pattern Sampling for Automatic Pattern Recognition. *Proc. IEEE*, 58(2):198-216.

Khazenie, N., and K. Richardson, 1991. Classification of Cloud Types Based on Spatial Textural Measures Using

NOAA-AVHRR Data. *Proc. IGARSS'91 Symp.*, 3:1701-1705.

Wechsler, H., 1980. Texture Analysis -- A Survey. *Signal Processing*, 2:271-282.

Welch, R. M., K. Kuo, and S. K. Sengupta, 1990. Cloud and Surface Textural Features in Polar Regions. *IEEE Trans. on Geoscience and Remote Sensing*, 28(4):520-528.

Work has also continued into special case GP algorithms: Integer, Zero–One, Fuzzy, Interactive and Chance–Constrained. A breakdown of publications in these areas is given in Romero [37]. In total he lists 355 papers dealing with GP applications in 26 distinct areas.

Research has been done to apply other Multi-Criteria and Management Science techniques to Goal Programming. These include interactive multi-criteria methods [38], 'Delphi' techniques [39, 40], Saaty's [41] analytical hierarchy approach [36, 23, 39], and resource planning and management systems(RPMS) networks [42]. Recently papers have been published dealing with some of the perceived 'errors' in G.P [37, 40, 43]. and explaining how these can be avoided by the correct setting of weights, goals, priority levels etc.

The remainder of the paper will be divided into four sections. Section 2 will deal with lexicographic(pre-emptive) GP, section 3 with weighted GP(non pre-emptive), section 4 with the connection between utility functions and GP, finally section 5 will draw conclusions as to the current direction of GP and the direction of the authors' future research.

2 Lexicographic GP

Of the 355 papers mentioned by Romero [37], 226 use the concept of Lexicographic GP(LGP), which requires the pre–emptive ordering of priority levels. The standard LGP model can be algebraically represented as:

$$Lex\ min\ \mathbf{a} = (g_1(\mathbf{n},\mathbf{p}), g_2(\mathbf{n},\mathbf{p}), ..., g_K(\mathbf{n},\mathbf{p}))$$

subject to,

$$f_i(\mathbf{x}) + n_i - p_i = b_i \quad i = 1, ..., m$$

This model has K priority levels, and m objectives. a is an ordered vector of these K priority levels.

A standard 'g' (within priority level) function is given by:

$$g_k(\mathbf{n},\mathbf{p}) = \alpha_{k_1} n_1 + ... + \alpha_{k_m} n_m + \beta_{k_1} p_1 + ... + \beta_{k_m} p_m$$

This paper will summarize the development of algorithms to solve the LGP model, work on the multi-dimensional dual [30, 44], and current thinking on methods of priority ranking and weighting within the priority levels. Some applications of LGP will be commented on, in an effort to outline which types of problem are suitable for an LGP approach, and which are better solved using other techniques.

3 Weighted GP

Weighted (or non-pre-emptive) GP(WGP) requires no pre-emptive ordering of the objective functions. Instead all the different deviations are placed in a single priority level objective with different weights to represent their importance.

Algebraically, a WGP has the following structure:

$$Min \ a = \sum_{i=1}^{k}(\alpha_i n_i + \beta_i p_i)$$

Subject to,

$$f_i(\mathbf{x}) + n_i - p_i = b_i \quad i = 1, \ldots.m$$
$$\mathbf{x} \in C_s$$

Where C_s is an optional constraint set. Of interest here are the problems caused by incommensurability, i.e. objective functions being measured in different units, and techniques used to overcome this. As in the LGP case, application areas will be outlined.

4 Utility Functions

The third section will deal with the connections between utility functions and the different types of GP. It will explore the literature on the problems caused in reconciling LGP and utility function theory. It will also examine recently developed techniques to model GP's more closely around their underlying objective functions [45].

5 Summary and Conclusions

The final section will draw conculsions as to the scope and limitations of GP and highlight areas in which the authors intend to conduct further research.

References

[1] CHARNES, A., COOPER, W.W., and FERGUSON, R. Optimal Estimation of Executive Compensation by Linear Programming, *Management Science*, 1, 138-151. 1955.

[2] CHARNES, A. and COOPER, W.W.. *Management Models and Industrial Applications of Linear Programming.* John Wiley and Sons, New York. 1961.

[3] IJIRI, Y. *Management Goals and Accounting for Control* . North Holland. Amsterdam. 1965.

[4] LEE, S.M. *Goal Programming for decision analysis.* Auerback, Philadelphia, 1972.

[5] IGNIZIO, J.P. *Goal Programming and Extensions.* Lexington, Mass.: Heath (Lexington Books), 1976.

[6] ALBRIGHT, S.C. Allocation of Research Grants to University Research Proposals. *Socio-Economic Planning Sciences*, 9, 189-195. 1975.

[7] JOINER, C. Academic Planning Through The Goal Programming Model *Interfaces*, **10**, 86-91. 1980.

[8] KILLOUGH, L.N. and SOUNDERS, T.L. A Goal Programming Model for Public Accounting Firms, *The Accounting Review*, **48**, 268-279. 1973.

[9] WHEELER, B.M. and RUSSELL, J.R.M. Goal Programming and Agricultural Planning, *Operational Research Quarterly*, **28**, 21-32. 1977.

[10] SAMOUILIDIS, J.E. and PAPPAS, I.A. A Goal Programming Approach to Energy Forecasting, *European Journal of Operational Research*, 5, 321-331. 1980.

[11] KUMAR, P.C., PHILIPPATOS, G.C., and EZZELL, J.R. Goal Programming and the Selection of Portfolios by Dual-purpose Funds. *Journal of Finance*, **33**, 303-310. 1979.

[12] CHISMAN J.A., and RIPPY, D. Optimal Operation of a Multipurpose Resorvior using Goal Programming, *The Clemson University Review of Industrial Management and Textile Science*, Fall, 69-82. 1977.

[13] HANNAN E.L. Allocation of Library Funds for Books and Standing Orders – a Multiple Objective Formulation, *Computers and Operational Research*, **5**, 109-114. 1978.

[14] DE KLUYVER C.A. An Exploration of Various Goal Programming Formulations–With Application to Advertising Media Scheduling, *Journal of the Operational Research Society*, **30**, 167-171. 1979.

[15] ZELENY, M. The Pros and cons of Goal Programming. *Computers and Operations Research*, **8**, 357–359, 1982.

[16] HARRALD, J., LEOTTA, J., WALLACE, W.A. and WENDELL, R.E. A Note on the Limitations of Goal Programming as Observed in Resource Allocation for Marine Enviromental Protection. *Naval Research Logistics Quarterly*, **25**, 733-739. 1978.

[17] DOBBINS, C.L. and MAPP, H.P. A Comparison of Objective Function Structures used in a Recursive Goal Programming–simulation Model of Farm Growth. *Southern Journal of Agricultural Economics*, **14**, 9-16. 1982.

[18] LARA, P. and ROMERO, C. An Interactive Multigoal Progamming Model for Determining Livestock Rations: an Application to Dairy Cows in Andalusia, Spain. *Journal of the Operational Research Society*, **43**, 945-953. 1992.

[19] NEAL, H.L., FRANCE, J., and TREACHER, T.T. Using Goal Programming in Formulating Rations for Pregnant Ewes, *Animal Production* , **42**, 97-104. 1986.

[20] MIN, H. A Model-Based Decision Support System for Locating Banks. *Information and Management*, **17**, 207-215. 1989.

[21] KWAK, N.K. and SCHNIEDERJANS, M.J. A Goal Programming Model for Selecting a Facility Location Site. *RAIRO-Operations Research*, 9, 1-14. 1985.

[22] GREENWOOD, A.G. and MOORE, L.J. An Inter-temporal Multi-goal Linear Programming Model for Optimizing University Tuitition and Fee Structures, *Journal of the Operational Research Society*, **38**, 599-613. 1987.

[23] DIMINNIE, C.B. and KWAK, N.K. A Hierarchical Goal-Programming Approach to Reverse Resource Allocation in Institutions of Higher Learning, *Journal of the Operational Research Society*, **30**, 59-66. 1986.

[24] FRANZ, L.S., BAKER, H.M., LEONG, G.K., and RAKES, T.R. A Mathematical Model for Scheduling and Staffing Multiclinic Health Regions, *European Journal of Operational Research*, **41**, 277-289. 1989.

[25] SALADIN, B.A. Goal Programming Applied to Police Patrol Allocation. *Journal of Operations Management*, **2**, 239-249. 1982.

[26] LEVARY, R.R. and AVERY, M.L. On the Practical Application of Weighting Equities in a Portfolio via Goal Programming, *Opsearch*, **21**, 246-261. 1984.

[27] BOOTH, G.G. and BESSLER, W. Goal Programming Models for Managing Interest Rate Risk, *Omega*, **17**, 81-89. 1989.

[28] IGNIZIO, J.P. Antenna Array Beam Pattern Synthesis via Goal Programming, *European Journal of Operational Research*, **6**, 286-290. 1981.

[29] O'GRADY, P.J., and MENON, U. A Multiple Criteria Approach for Production Planning of Automated Manufacturing, *Engineering Optimization*, **8**, 161-175. 1985.

[30] IGNIZIO, J.P. An Algorithm for Solving the Linear Goal Programming Problem by Solving its Dual, *Journal of the Operational Research Society*, **36**, 507-515. 1985.

[31] SCHNIEDERJANS, M.J. and KWAK, N.K. An Alternative Solution Method for Goal Programming Problems : A Tutorial, *Journal of the Operational Research Society*, **33**, 247-251. 1982.

[32] ARTHUR, J.A. and RAVINDRAN, A. An Efficent Goal Programming Algorithm Using Constraint Partitioning and Variable Elimination. *Management Science*, **24**, 1109-1119, 1978.

[33] LEE, S.M. and LUEBBE, R.L. A Zero-One Goal-Programming Algorithm Using Partitioning and Constraint Aggregation. *Journal of the Operational Research Society*, **38**, 633-640, 1987.

[34] MARKLAND, R.E. and VICKERY, S.K. The Efficent Computer Implementation of a Large-Scale Integer Goal Programming Model. *European Journal of Operational Research*, **26**, 341-354, 1986.

[35] OLSON, D. A Comparison of Four Goal Programming Algorithms. *Journal of the Operational Research Society*, **35**, 347-354, 1984.

[36] GASS, S.I., A Process for Determining Priorities and Weights for Large-Scale Linear Goal Programmes. *Journal of the Operational Research Society*, **37**, 779-785, 1986.

[37] ROMERO, C. *Handbook of Critical Issues in Goal Programming*. Pergamon Press, 1991.

[38] MASUD, A.S. and HWANG, C.L. Interactive Sequential Goal Programming, *Journal of the Operational Research Society*, **32**, 391-400, 1981.

[39] KHORRAMSHAGOL, R. and AZANI, H. A Decision Support System for Effective Systems Analysis and Planning, *Journal of Information and Optimization Sciences*, **9**, 41-52, 1988.

[40] KHORRAMSHAGOL, R. and HOOSHIARI, A. Three Shortcomings of Goal Programming and their Solutions. *Journal of Information and Optimization Sciences*, **12**, 459-466, 1991.

[41] SAATY, T.L. *The Analytical Hierarchy Process*, McGraw-Hill International, New York, 1981.

[42] SHIM, J.P. and CHIN, S.G. Goal Programming: The RPMS Network Approach, *Journal of the Operational Research Society*, **42**, 83-93, 1991.

[43] MIN, H. and STORBECK, J. On the Origin and Persistence of Misconceptions in Goal Programming, *Journal of the Operational Research Society*, **42** , 301-312, 1991.

[44] IGNIZIO, J.P. *Linear Programming in Single and Multiple Objective Systems.*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1982

[45] MARTEL J.M. and AOUNI B. Incorporating the Decision-maker's Preferences in the Goal-programming Model. *Journal of the Operational Research Society*, **41**, 1121-1132, 1990.

MULTI-STAGE ECONOMIC LOT SCHEDULING PROBLEM

Ayşegül Toker Terzi and Nesim Erkip

Department of Industrial Engineering
Middle East Technical University
06531, Ankara, Turkey

The Economic Lot Scheduling Problem (ELSP) is to economically schedule lots of one or more products on a single machine. Demand is constant, backlogging is not allowed and the planning horizon is infinite. The problem is to minimize total operating cost per unit time which is comprised of setup costs and inventory costs. Setup costs are incurred whenever a production for a lot is begun and inventory carrying costs can be defined as the time value of money tied up in inventory.

An extension to single machine/facility problem is the study of environments where products are manufactured through several operations. Such systems are, in general, called as multi-stage production systems. Multi-stage production systems received a lot of academic attention in recent years focussing on the control of work-in-process inventory and its functional relationship to the manufacturing cycle time. It is a very well known fact by now, the larger the production lot size, the longer the manufacturing cycle, which in turn, increases the work-in-process inventory. There exists a vast literature modelling this relationship to varying degrees in different models for different system configurations.

The Multi-stage Economic Lot Scheduling Problem (MS-ELSP) brings together two important problem characteristics inherent to multi-item and multi-stage problems. In a multi-item problem, the main issue is that of creating schedules which avoids the interference that is likely to occur when two or more products compete for the same facility. We will refer to this as the "feasibility issue". In a multi-stage environment, the production should be synchronized so that concurrent production of the same lot is not possible in the consecutive stages. This characteristic leads to the definition of work-in-process inventory which, in fact, is a tool for the synchronization of production among stages. Thus, in multi-stage problems, creating schedules owing this property will be referred to as "consistency issue". This study addresses the Multi-stage Economic Lot Scheduling Problem with the objective of determining feasible and consistent schedules which result from the conventional tradeoff between setup costs and inventory holding costs comprising the total cost of a schedule.

In this research, we restrict the study of MS-ELSP to serial systems where there are m products to be manufactured through n distinct stages. We first analyze the two product - two stage problem. In order to guarantee feasibility, common cycle solutions in which the possible values of cycle times for all items are constrained to a single cycle time value, T, are sought for. In a two-stage production system, production of a lot on the second stage cannot begin until its production on the first stage is completed. Therefore, production between stages should be synchronized so that we end up with consistent schedules. To ensure consistency, we define a constraint

for each product which also provide information about the work-in-process inventories.

Another important point in this study is the presence of nonnegative setup times. Setup times mean a loss in the productive capacity and their effect on lot sizes is the most significant when the capacity utilization is high. On the other hand, work-in-process inventories tend to increase with increasing capacity utilizations. Therefore, ignorance of setup times will result in overestimated lot sizes due to underestimation of work-in-process inventories.

The mathematical programming formulation of this problem is developed where the objective function is nonlinear with a linear set of constraints. Setting the cycle time to a fixed value, we first linearize the objective function. By using the dual problem and complementary slackness, the optimal solution of this problem and thus the optimal cycle time for the two product - two stage problem are obtained. Besides, we have the exact terms for the work-in-process inventories (queueing inventories: inventory that built up on the previous stage if the successor stage is busy with processing the other products) since they can be expressed explicitly as analytical functions of the cycle time. Then, we generalize our result to multi-product case in a two stage system which constitutes a basis for the analysis of the m-product, n-stage economic lot scheduling problem.

Key words: Economic Lot Scheduling Problem, multi-stage

COMPUTATIONAL EXPERIENCE WITH A PRIMAL-DUAL INTERIOR-POINT METHOD FOR SMOOTH CONVEX PROGRAMMING

J.-P. Vial*
Département d'Ecomomie Commerciale et Industrielle
Université de Genève
102 Bd Carl Vogt, CH-1211 Genève 4, Switzerland

August 1992

Key words: Interior-point method, Primal-dual algorithm, Convex programming, Computational experience.

*This research has been supported by the Fonds National de la Recherche Scientifique, grant #12 – 26434.89.

In this paper we present computational experience with a primal-dual interior point for smooth convex programming problems of the type

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \\ & g(x) \leq 0, \end{array} \qquad (1)$$

where $c \in \mathbf{R}^n$ and $g : \mathbf{R}^n \to \mathbf{R}^m$ is a vector-valued function. We assume that each component g_i is convex. Let $s \in \mathbf{R}^m$, be the vector of slack variables. The inequality constraints in (1) are replaced by

$$g(x) + s = 0, \quad s \geq 0.$$

Given a parameter $\mu > 0$, we associate with (1) the barrier problem

$$\begin{array}{ll} \min & c^T x - \mu \sum_{i=1}^{m} \ln s_i \\ \text{s.t.} & \\ & g(x) + s = 0 \\ & s \geq 0. \end{array} \qquad (2)$$

We assume that Slater's condition holds:

Assumption 0.1 *There is an $x \in \mathbf{R}^n$ such that $g(x) < 0$.*

We also assume

Assumption 0.2 *The set $\{x : g(x) \leq 0$ and $c^T x \leq \theta\}$ is bounded for all θ.*

Under these assumptions *Problem* (2) has a solution. The necessary and sufficient conditions for optimality, namely the Karush-Kuhn-Tucker equations, or KKT equations, are

$$\begin{aligned} Ys - \mu e &= 0 && (3) \\ g(x) + s &= 0 && (4) \\ \left(\frac{\partial g}{\partial x}\right)^T y + c &= 0, && (5) \end{aligned}$$

with $s \geq 0$ and $y \geq 0$. Here

$$\frac{\partial g}{\partial x} = \left\{ \frac{\partial g_i(x)}{\partial x_j} \right\}$$

is the Jacobian matrix of g and $y \in \mathbf{R}^m$ is a vector of dual variables.

Let

$$F : \mathbf{R}^m \times \mathbf{R}^m \times \mathbf{R}^n \to \mathbf{R}^m \times \mathbf{R}^m \times \mathbf{R}^n$$

be a multi-valued function defined by

$$F(z) = \begin{pmatrix} F_c \\ F_p \\ F_d \end{pmatrix} = \begin{pmatrix} Ys - \mu e \\ \frac{\partial L}{\partial y} \\ \frac{\partial L}{\partial x} \end{pmatrix},$$

with $z = (y, s, x)$. F also depends on the parameter $\mu > 0$. With this notation, the KKT system is simply $F(z) = 0$.

We also introduce the Lagrangean

$$L(y; x, s) = c^T x + y^T (g(x) + s). \qquad (6)$$

The KKT system (3) - (5) can be rewritten as

$$Ys - \mu e = 0, \quad \frac{\partial L}{\partial x} = 0, \quad \text{and} \quad \frac{\partial L}{\partial y} = 0.$$

Following usual terminology, a point $z = (y, s, x)$ is interior if $y > 0$ and $s > 0$. We do not require it to be primal or dual feasible. At such a point, we define the Newton direction $dz = (dy, ds, dx)$ by

$$\frac{\partial F}{\partial z} dz + F = 0. \tag{7}$$

Note that

$$\frac{\partial F}{\partial z} = \begin{pmatrix} S & Y & 0 \\ 0 & \frac{\partial^2 L}{\partial y \partial s} & \frac{\partial^2 L}{\partial y \partial x} \\ \frac{\partial^2 L}{\partial x \partial y} & 0 & \frac{\partial^2 L}{\partial x^2} \end{pmatrix},$$

with

$$\frac{\partial^2 L}{\partial y \partial s} = I, \quad \frac{\partial^2 L}{\partial y \partial x} = \frac{\partial g}{\partial x}, \quad \frac{\partial^2 L}{\partial x \partial y} = \left(\frac{\partial g}{\partial x}\right)^T, \quad \text{and} \quad \frac{\partial^2 L}{\partial x^2} = \sum_{i=1}^{m} y_i \frac{\partial^2 g_i}{\partial x^2}.$$

Since the g_i are convex, $\frac{\partial^2 L}{\partial x^2}$ is positive semi-definite. Let us make the further assumption

Assumption 0.3 *Let $y > 0$ and $s > 0$. The matrix*

$$H := \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial x \partial y} Y S^{-1} \frac{\partial^2 L}{\partial y \partial x}$$

is positive definite.

A sufficient condition for that is:

$$\frac{\partial^2 L}{\partial x^2} = \sum_{i=1}^{m} y_i \frac{\partial^2 g_i}{\partial x^2}$$

is positive definite, or $\frac{\partial g}{\partial x}$ has full row rank, or both.

Under Assumption 0.3, $\frac{\partial F}{\partial z}$ is regular at any interior point. Thus

$$dz = -\left(\frac{\partial F}{\partial z}\right)^{-1} F.$$

Let us explicitly write and solve the system (7) in dy, ds and dx:

$$\begin{aligned} S dy + Y ds + F_c &= 0 \\ ds + \frac{\partial^2 L}{\partial y \partial x} dx + F_p &= 0 \\ \frac{\partial^2 L}{\partial x \partial y} dy + \frac{\partial^2 L}{\partial x^2} dx + F_d &= 0. \end{aligned}$$

In these expressions we used the fact that $\frac{\partial^2 L}{\partial y \partial s} = I$.

The algorithm goes as follows: Given an interior, but not necessarily feasible, point, we compute the search direction dx associated with μ. Then a step is taken along that direction such that the interior property is maintained. Namely, let $\bar{\alpha} := \max\{\alpha : y + \alpha dy \geq 0, s + \alpha ds \geq 0\}$ and let $0 < \gamma < 1$. Then the next iterate is given by

$$\begin{aligned} x &:= x + \gamma\alpha dx \\ s &:= s + \gamma\alpha ds \\ y &:= y + \gamma\alpha dy. \end{aligned}$$

Te choice of μ is adaptive. For "normal" steps, we take $\mu = \frac{y^T s}{m^2}$. If $\min y_i s_i \leq \gamma \frac{y^T s}{m}$, the vector Ys is considered excessively unbalanced and we take $\mu = \frac{y^T s}{m}$. This step is named "centering".

We tested our algorithm on a sample of medium size random problems. We primarily studied the effect of varying the size of the problems. We observed that the number of iterations increases slowly with the number of constraints and, surprisingly enough, it decreases with the number of free variables in the case of quadratically constrained problems.

We analyzed the influence of centering and showed it to be positive. We also studied alternative strategies for the step size. It turns out that taking a fixed fraction of the maximal step size works well in practice. Moreover the fraction can be extremely close to 1 without any negative effect on the performance of the method. Finally, we looked at different choices for the starting point.

We applied this algorithm to linear programming problems. The algorithm behaves a bit differently than with quadratic constraints. The iteration count increases both with the number of constraints and the number of free variables. For the former the increase is slower. The figures are reasonable.

It Is Difficult to Find a Difficult Problem for Scheduling of Identical Parallel Machines

Béla Vizvári and Ramazan Demir
Bilkent University, 06533 Bilkent, Ankara, Turkey, VIZVARI@TRBILUN.BITNET

1 INTRODUCTION

Mathematical programming and theory of scheduling have a lot of optimization problems which are NP-hard in spite of their very simple structure. Thus these problems are considered to be difficult to solve. But some of them are easy in the sense that there are straightforward ways to generate feasible solutions of them, e.g. the knapsack problem, the TSP problem and many scheduling problems.

One of them is the scheduling of identical parallel machines where the maximal completion time has to be minimized. This problem is the topic of this experimental study. It has several heuristics. The two basic ones are Graham's list scheduling and the multi-fit algorithm. There are known upper bounds for the relative accuracy of the heuristic solutions provided by these methods. The two algorithms have quite different strategies. This is the reason that some problems worst from the point of view of list scheduling can be solved exactly by the multi-fit algorithm and vice versa. This gives the question that *how bad accuracy can have the better of the list scheduling and the multi-fit solutions*. This was the initial question of this research. Another algorithm called interchanging method has been also investigated. The research made necessary to sharpen the well-known lower bound of the optimal value of the objective function, too.

2 THE SCHEDULING PROBLEM

In the classical problem of the scheduling of parallel machines n jobs have to be distributed among m identical machines in such a way that the makespan is minimal.

The whole operation starts at time 0. The machine independent processing times are denoted by $p_j (j = 1, ..., n)$ which are positive integers. It is easy to see that there is at least one optimal solution such that the machines start to work at t=0 and are working without any idle time until all jobs assigned to them have been finished.

Let C_j be the completion time of job j. The maximal completion time, i.e. $\max\{C_j : j = 1, ..., n\}$, is denoted by C^*.

Theorem 1 *[Graham 69], [Coffman et al. 78] In any problem*

$$\begin{gathered}\max\{\tfrac{1}{m}\textstyle\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\} \\ \leq C^* \leq \\ \max\{\tfrac{2}{m}\textstyle\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}.\Box\end{gathered} \qquad (1)$$

The interval in which the optimal value must lie is denoted by $[L, U]$, i.e.

$$L = \lceil \max\{\frac{1}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}\rceil \qquad (2)$$

and

$$U = \lfloor \max\{\frac{2}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}\rfloor. \qquad (3)$$

Both the list scheduling and the multi-fit algorithm start with the determination of the nonincreasing order of the processing times. The two algorithms assign the jobs to machines in that order. Therefore without loss of generality it may be assumed that

$$p_1 \geq p_2 \geq ... \geq p_n \qquad (4)$$

The rule of the list scheduling is that

every job is assigned to a machine having minimal current load. (LS)

Theorem 2 *[Graham 69] Let $C(LS)$ be the value of the solution provided by the list scheduling. Then*

$$\frac{C(LS)}{C^*} \leq \frac{4}{3} - \frac{1}{3m}.\square \qquad (5)$$

Theorem 3 *[Graham 69] If there is an optimal solution which assigns to each machine at most 2 jobs, then the solution given by the list scheduling is optimal.* □

The multi-fit algorithm consists of two parts. A greedy method is the internal part and a logarithmic search is the external part which organizes the applications of the greedy method. For the internal part an upper bound K of the optimal value is assumed. The greedy method assigns each job to the first machine into it fits not exceeding the upper bound K. In the external part a current lower bound and a current upper bound are assumed and are denoted by lc and uc. For the internal part K is chosen as $\frac{lc+uc}{2}$. If the greedy method was able to find a solution not worst then K, then uc becomes $\lfloor K \rfloor$, otherwise $lc = \lceil K \rceil$. The process is repeated until the condition

$$uc = lc$$

is not satisfied. Notice that it follows from the assumption of the integrality of the processing times that the number of applications of the greedy method is $O(log(U - L))$. Thus the multi-fit algorithm is polynomial.

Theorem 4 *[Friesen 84] Let $C(MF)$ be the value of the solution provided by the multi-fit algorithm. Then*

$$\frac{C(MF)}{C^*} \leq 1.2 \,\square \qquad (6)$$

A third heuristic method called interchanging algorithm has been applied in this research. It makes the following steps starting from any solution. It interchanges one job of the most loaded machine with one job of another machine. The interchange is possible if and only if the maximal completion time is decreased in this way. Let s and t, resp., be indices of the most loaded and the other machines, resp. The current load of the machines are denoted by L_s and L_t. Suppose that the job i of machine s is interchanged with job j. Then the following two conditions must hold

$$p_i > p_j \tag{7}$$

and

$$L_t + p_i - p_j < L_s. \tag{8}$$

In the current version if a possible interchange is found then it has been executed. The order of checking Conditions (7) and (8) is as follows. The jobs of the most loaded machine are compared with the jobs of another machine taking the other machines in an increasing load order. The jobs of the two machines are taken in a decreasing processing time order. One jc of the most loaded machine is compared with all of the jobs of the other machine. If no possible interchange is found then the next job of the most loaded machine is taken. The number of comparisons of one iteration are $O(n^2)$. To get a polynomial algorithm the number of interchanges has been limited by $m+2$. In the current version the solution provided by list scheduling is the starting point. This algorithm is one of simplest possible interchanging methods. In more general a subset of jobs can be interchanged for another subset of jobs. In that case the complexity of the selection of the two subsets is much higher.

3 IMPROVEMENTS OF THE LOWER BOUND

The randomly generated problems have not been solved with any kind of enumerative methods. One easy way to prove the optimality of a solution is that the value of it and the lower bound coincide. Therefore it was important to find some ways to improve the lower bound.

In (2) only two information are taken into consideration, the average load and the maximal processing time. The following two sharpening of the lower bound are based on the fact that what is the number of jobs which must be assigned to certain machines.

Theorem 5 *Assume that (4) holds. Then*

$$C^* \geq p_{n-\lceil \frac{n}{m} \rceil +1} + \dots + p_n \quad \square \tag{9}$$

Theorem 6 *Assume that (4) holds. Let*

$$r = n - \left\lfloor \frac{n}{m} \right\rfloor m$$

. If $r > 1$

$$C^* \geq \min\left\{ \frac{\sum_{j=n-\lceil \frac{n}{m} \rceil r+1}^{n} p_j}{r},\ \sum_{j=n-\lceil \frac{n}{m} \rceil}^{n} p_j \right\}. \ \square \tag{10}$$

In some cases there are jobs which are not effecting C^*, because their processing times are relatively very small. In that cases the following observation is useful.

Theorem 7 *Let S be any subset of the jobs. Let L^S be any lower bound for the problem defined by the jobs those in S. Then L^S is a lower bound for the original problem.* □

Theorem 8 *Let k be any index with $1 \leq k \leq n$. Assume that: (i) the list scheduling has assigned until that point exactly k jobs to machines, (ii) none of the machines has more than two jobs, (iii) $p_{k-2} + p_{k-1} + p_k$ is at least as great as the current load of any machine. Then the current maximal load is a lower bound for the optimal value of the problem.* □

Theorem 9 *Let k be a fixed index and*

$$t_j = \left\lfloor \frac{p_j}{p_k} \right\rfloor \quad j = 1, \ldots, n.$$

Then

$$\left\lceil \frac{\sum_{j=1}^{n} t_j}{m} \right\rceil p_k \leq C^*.$$

4 COMPUTATIONAL EXPERIENCES

The computational experiences have been made in three phases. In the first phase about 500.000 problems belonging to different classes have been generated. In this phase some observations have been made which modified the objectives of the research. The second phase was the main one in which 1.200.000 problems have been generated in a wide range of problem classes to find difficult problems. Further attempts have been made to find more difficult problems in the most hopeful problem classes.

Definition 1 *Let $C(LS)$ and $C(MF)$ and $C(IC)$ and C^* be, resp., the value of the solution provided by the list scheduling and the multi-fit algorithm and the interchanging method and of the optimal solution, resp. A particular problem is called* first order difficult *if the value*

$$\frac{\min\{C(LS), C(MF)\}}{C^*} \tag{11}$$

is high. It is called second order difficult *if the value*

$$\frac{\min\{C(LS), C(IC), C(MF)\}}{C^*} = \frac{\min\{C(IC). C(MF)\}}{C^*} \tag{12}$$

is high.

This definition is not correct in a strict mathematical sense, because the meaning of the word "high" is undefined. This meaning has been determined during the experiences.

A problem class is determined by the following parameters: m - the number of machines, n - the number of jobs, p - the maximal possible processing time; the processing times are generated randomly by the $[1,p]$ integer uniform distribution.

In the experiences the following formulas have been used instead of (11) and (12)

$$\frac{\min\{C(LS), C(MF)\}}{\hat{L}} \tag{13}$$

and

$$\frac{\min\{C(IC), C(MF)\}}{\hat{L}} \tag{14}$$

where $\hat{L}$ is some lower bound of the optimal value of the objective function.

4.1 Observations of the First Phase

In the first phase only the list scheduling and the multi-fit algorithm have been used.

At the beginning of the experiences L has been chosen as $\hat{L}$. Some problems seemed to be difficult although an optimal solution has been obtained by one of the methods. In some cases this fact could be proven by one of the improvements of the lower bound discussed in Section 3.

Some problems had just the opposite behaviour. Here the lower bound coincided with the optimal value. In many cases this fact could be proven by the interchanging algorithm. This is the reason that this method had to be involved into the investigations.

Among the most difficult problems found in this phase there were many such that the smallest processing time was relatively great. Therefore in the second phase of the experiences the generation of the the problems has been modified as follows. The first thousand problems has been generated as earlier. In the case of the problems of the second thousand the processing times were increased by 1, in the case of the third thousand by 2, e.t.c. This cannot be applied for all of the classes, because in some cases if the increase is not less than a certain value, the problem regardless the generated random numbers becomes trivial.

The problems which seemed to be difficult were belonging to two different categories. The first one is the set of first order difficult problems. The most difficult problem in this sense was the following. $n = 10$, $m = 3$ and the processing times are 30, 29, 24, 18, 17, 17, 17, 14, 13, 13. The solution provided by the list scheduling is as follows: M1: 30, 17, 13; M2: 29, 17, 14; M3: 24, 18, 17, 13. The multi-fit solution is: M1: 30, 29, 13; M2: 24, 18, 17, 13; M3: 17, 17, 14. Both of them have the value 72. But the optimal solution is the following: M1: 30, 17, 17; M2: 29, 18, 17; M3: 24, 14, 13, 13. The value of it is 64. Since that time Definition 1 has had the meaning that a problem is first order difficult if

$$\frac{\min\{C(LS),C(MF)\}}{C^{*}} > \frac{9}{8}.$$

The computationally difficult problems belong to the second category. In the case of such a problem it is difficult either to find the optimal solution or to prove the optimality of a solution generated by one of the heuristics.

4.2 Experiences of the Main Phase

In the second phase an intensive search has been carried out for difficult problems. 100.000 problems have been generated in each of the problem classes. The generated solutions are within 105% and even 101% of the improved lower bound in the case of a very great part of the problems in each class. These results are summarized in Table 1.

It turned out that none of the list scheduling and the multi-fit algorithm is superior to the other one. This is indicated by the numbers of problems such that the appropriate heuristic solution is within 101%. The number of problem classes for which a method is superior to the other one is approximately is the same for both algorithms. The behaviour of both methods are very different in the different classes. But the "the better of list scheduling and multi-fit" seems to be much stable.

n/m/p	101%	101%	105%	105%
	LS-MF	IC-MF	LS-MF	IC-MF
10/3/15	88067	94179	98817	99897
15/3/15	95177	99441	99997	99999
10/3/30	73970	85831	97418	99815
15/3/30	89662	99002	99999	100000
10/3/60	45787	69949	94304	99582
15/3/60	80167	98599	99996	100000
30/3/15	99910	100000	100000	100000
30/3/30	99917	100000	100000	100000
30/3/60	99132	100000	100000	100000
10/5/15	98244	98245	99043	99044
20/5/15	89275	97461	99995	100000
60/5/60	100000	100000	100000	100000
$\sum$	960051	1043707	1089569	1098337
percentage	87.27	94.88	99.05	99.85

Table 1: The numbers of problems having good heuristic solution

parameters	MF	LS
10/3/15	87795	1318
15/3/15	7324	93662
10/3/30	73529	950
15/3/30	12688	86078
10/3/60	44898	1568
15/3/60	23804	72245
30/3/15	26577	99725
30/3/30	50626	99591
30/3/60	42090	99132
10/5/15	98236	97997
20/5/15	3649	87973
60/5/60	100000	99111

Table 2: Comparison of the list scheduling and multi-fit heuristics

The most first order difficult problem which has been found in this phase is the following. $n = 10$, $m = 3$ and the processing times are 15, 14, 12, 9, 8, 8, 8, 7, 6, 6. The solution provided by the list scheduling is as follows: M1: 15, 8, 6, 6; M2: 14, 8, 7; M3: 12, 9, 8. The multi-fit solution is: M1: 15, 14, 6; M2: 12, 9, 8, 6; M3: 8, 8, 7. Both of them have the value 35. But the optimal solution is the following: M1: 15, 8, 8; M2: 14, 9, 8; M3: 12, 7, 6, 6. The value of it is 31.

4.3 Further difficult Problems

The aim of the third phase has been to find further difficult problems. Some new problem classes are introduced, because it is likely on the basis of the previous experiences that these classes contain the desired items. At the end of this phase the number of the generated problems have exceeded 2.000.000.

The class 19/8/15 contained the known most difficult problem. The processing times of it are: 21, 21, 20, 20, 19, 18, 17, 17, 16, 16, 16, 16, 12, 12, 12, 11, 11, 10, 10. The multi-fit solution is: M1: 21, 21; M2: 20, 20; M3: 19, 18; M4: 17, 17; M5: 16, 16, 10; M6: 16, 16, 10; M7: 12, 12, 12; M8: 11, 11. The value of it is 42, which is achieved at M1 and M5 and M6. The solution provided by the list scheduling with value 43 is this: M1: 21, 11, 11; M2: 21, 12; M3: 20, 12, 10; M4: 20, 12, 10; M5: 19, 16; M6: 18, 16; M7: 17, 16; M8: 17, 16; In the optima; solution the completion time is 37 on all of the machines except the last one where it is 36: M1: 21, 16; M2: 21, 16; M3: 20, 17; M4: 20, 17; M5: 19, 18; M6: 16, 11, 10; M7: 16, 11, 10; M8: 12, 12, 12.

The development of the accuracy of the most known first order difficult problems has been: $\frac{9}{8} < \frac{35}{31} < \frac{42}{37}$. The value 42/37, which in not proven to be an upper bound, is less than the value 72/61 guaranteed by the algorithm of [Friesen-Langston 86], which uses many operations from a practical point of view.

There was no improvement in the position of most second order difficult problem in this phase.

4.4 Some Other Observations

Some other observations are obtained from the experiences. An important one is the following. If $L \neq U$ then U is far from the optimal value. The 10/5/15 class is the only one where the ratio (13) had a value greater then 1.22. The observed greatest value is 1.42.

The aim of the improvements of the lower bound was to decrease the number of cases to check. In Table 5 the the number of problems which have been proved to be solved within 101%, and the observes worst (14) ratio observed before any improving and after improving (without the application of Theorem 9) are provided for the better of multi-fit and interchanging procedure.

parameters	101%		(14)	
	before	after	before	after
10/3/15	88078	89871	1.217	1.120
15/3/15	99344	99344	1.030	1.030
10/3/30	65089	68313	1.262	1.102
15/3/30	99098	99098	1.032	1,032
10/3/60	44046	71572	1.211	1.100
15/3/60	99045	99045	1.032	1.032
30/3/15	100000	100000	1.005	1.005
30/3/30	100000	100000	1.007	1.007
30/3/60	100000	100000	1.005	1.005
10/5/15	37056	88469	1.412	1.200
20/5/15	97240	97240	1.040	1.040
60/5/60	100000	100000	100000	100000

Table 3: The effect of the improvements of the lower bound

References

[Coffman et al. 78] Coffman, E.G.Jr., Garey, M.R., Johnson, D.S., An application of bin-packing to multiprocessor scheduling, SIAM J. Comput., 7(1978) 1-17.

[Friesen 84] Friesen, D.K., Tighter bounds for the multifit processor scheduling algorithm, SIAM J. Comput., 13(1984), 170-181.

[Friesen-Langston 86] Friesen, D.K., Langston, M.A., Evaluation of a MULTIFIT-based scheduling algorithm, J. Algorithms, 7(1986), 35-59.

[Graham 69] Graham, R.L., Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17(1969) 416-429.

Concepts for Parallel Tabu Search

Stefan Voß
Technische Hochschule Darmstadt, FB 1 / FG Operations Research,
Hochschulstraße 1, D - 6100 Darmstadt, Germany

Abstract

Tabu Search is a metastrategy for guiding known heuristics to overcome local optimality. Successful applications of this kind of metaheuristic to a great variety of problems have been reported in the literature. Recently some implementations of tabu search on parallel computers have come up. Whereas these implementations are tailored to specific problems we attempt to provide ideas for a more general concept for developing parallel tabu search algorithms.

1 Introduction

Due to the complexity of a great variety of combinatorial optimization problems, heuristic algorithms are especially relevant for dealing with large scale problems. The main drawback of algorithms such as deterministic exchange procedures is their inability to continue the search upon becoming trapped in a local optimum. This suggests consideration of recent techniques for guiding known heuristics to overcome local optimality. Following this theme, the application of the *tabu search* metastrategy for solving combinatorial optimization problems is investigated.

The key issue in designing parallel algorithms is to decompose the execution of the various ingredients of a procedure into processes executable by parallel processors. Improvement procedures like tabu search or simulated annealing at first glance, however, have an intrinsic sequential nature due to the idea of performing the neighbourhood search from one solution to the next. Therefore, there is not yet a common or generally applicable parallelization of tabu search in the literature. In the sequel we attempt to describe some general ideas and a classification scheme for parallel tabu search algorithms.

In Section 2, we present an outline of tabu search. Before describing some concepts for parallel tabu search algorithms in more detail (see Section 4), we briefly discuss some of the common parallel machine models and algorithms in Section 3. Some examples are given in Section 5 and finally some conclusions are drawn (Section 6). The attempt, of course, is not to give a complete treatment of parallel tabu search but to sketch the potential this area of research carries. For a more detailed treatment of the ideas of this paper see Voß (1992).

2 Tabu Search

Many solution approaches are characterized by identifying a neighbourhood of a given solution which contains other (*transformed*) solutions that can be reached in a single iteration. A transition from a feasible solution to a transformed feasible solution is referred to as a *move* and may be described by a set of one or more *attributes*. In a zero-one integer programming context, e.g., these attributes may be the set of all possible value assignments or changes in such assignments for the binary variables. (Then two attributes denoting that a certain binary variable is set to 1 or 0, may be called *complementary* to each other.) Following a steepest descent/mildest ascent approach, a move may either result in a best possible improvement or a least deterioration of the objective function value. Without additional control, however, such a process can cause a locally optimal solution to be re-visited immediately after moving to a neighbour.

To prevent the search from endlessly cycling between the same solutions, tabu search may be visualized as follows. Imagine that the attributes of all moves are stored in a *running list*, representing the trajectory of solutions encountered. Then, related to a sublist of the running list a so-called *tabu list* may be defined. Based on certain restrictions, it keeps some moves, consisting of attributes complementary to those of the running list, which will be forbidden in at least one subsequent iteration because they might lead back to a previously visited solution. Thus, the tabu list restricts the search to a subset of admissible moves (consisting of admissible attributes or combinations of attributes). This hopefully leads to 'good' moves in each iteration without re-visiting solutions already encountered. A general outline of a tabu search procedure (for solving a minimization problem) may be described as follows:

Tabu Search

```
Given: A feasible solution $x^*$ with objective function value $z^*$.
Start: Let $x := x^*$ with $z(x) = z^*$.
Iteration:
while stopping criterion is not fulfilled[1]do begin
    (1)  select best admissible move that transforms $x$ into $x'$ with objective function value $z(x')$ and add its attributes to the running list
    (2)  perform tabu list management: compute moves to be set tabu, i.e., update the tabu list
    (3)  perform exchanges: $x := x', z(x) = z(x')$
         if $z(x) < z^*$ then $z^* := z(x), x* := x$ endif
endwhile
Result: $x^*$ is the best of all determined solutions, with objective function value $z^*$.
```

* * *

For a background on tabu search and a number of references on successful applications of this metaheuristic see, e.g., Glover (1989, 1990), Glover and Laguna (1992), and Voß (1992).

[1] A possible stopping criterion can be, e.g., a prespecified time limit.

Tabu List Management

Tabu list management concerns updating the tabu list, i.e., deciding on how many and which moves have to be set tabu within any iteration of the search. Up to now, the most popular approach in literature is to apply static methods like the *tabu navigation method* (TNM).

In TNM, single attributes are set tabu as soon as their complements have been part of a selected move. The attributes stay tabu for a distinct time, i.e. number of iterations, until the probability of causing a solution's re-visit is small. The efficiency of the algorithm depends on the choice of the tabu status duration, i.e. the length tl_size of the tabu list. (In the literature often a 'magic' tl_size=7 is proposed.) For the sake of an improved effectivity, a so-called *aspiration level criterion* is considered, which permits the choice of an attribute even when it is tabu. This can be advantageous when a new best solution may be calculated, or when the tabu status of the attributes prevent any move from feasibility.

The static approach, though successful in a great number of applications, seems to be a rather limited one. Another probably more fruitful idea is to define an attribute as being *potentially tabu* if it belongs to a chosen move and to handle it in a *candidate list* first. Via additional criteria these attributes can be definitely included in the tabu list if necessary, or excluded from the candidate list if possible. Therefore, the candidate list is an intermediate list between a running list and a tabu list. Glover (1990) suggests the use of different candidate list strategies in order to avoid extensive computational effort without sacrificing solution quality. In the sequel, we sketch the following dynamic strategies for managing tabu lists: the *cancellation sequence method* (CSM, in a revised version, cf. Dammeyer et al. (1991)), and the *reverse elimination method* (REM).

CSM as well as REM both use additional criteria for setting attributes tabu. The primary goal is to permit the reversion of any attribute but one between two solutions to prevent from re-visiting the older one. To find those *critical* moves, CSM needs a candidate list that contains the complements of attributes being potentially tabu. This *active tabu list* (ATL) is built like the running list where elimination of certain attributes is furthermore permitted. Whenever an attribute of the last performed move finds its complement on ATL this complement will be eliminated from ATL. All attributes between the cancelled one and its recently added complement build a *cancellation sequence* separating the actual solution from the solution that has been left by the move that contains the cancelled attribute. Any attribute but one of a cancellation sequence is allowed to be cancelled by future moves. This condition is sufficient but not necessary, as some additional aspects have to be taken into account so that CSM works well.

The method works well for the case that a move consists of exactly one attribute, i.e., when so-called *single-attribute moves* are considered instead of *multi-attribute moves*. In addition, the corresponding parameters have to be chosen appropriately (e.g. the tabu list duration of a tabu attribute, and how to apply the aspiration level criterion). Applying CSM to multi-attribute moves needs additional criteria to prevent errors caused by uncovered special cases. E.g. for *paired-attribute moves* (moves consisting of exactly two attributes) those moves must be prohibited that may cancel a cancellation sequence consisting of exactly two attributes (because none of them is tabu when choosing a move). In addition, for building a cancellation sequence, the remaining attributes of the older and the current move are not necessarily taken into consideration. This depends on the order in which the move's attributes are added to ATL.

The conditions of TNM and CSM need not be necessary to prevent from re-visiting previously encountered solutions. Necessity, however, can be achieved by REM. The idea of REM is that any solution can only be re-visited in the next iteration if it is a neighbour of the current solution. Therefore, in each iteration the running list will be traced back to determine all moves which have to be set tabu (since they would lead to an already explored solution). For this purpose, a *residual cancellation sequence* (RCS) is built up stepwise by tracing back the running list. In each step exactly one attribute is processed. from last to first. After initializing an empty RCS, only those attributes are added whose complements are not in the sequence. Otherwise their complements in the RCS are eliminated (i.e. cancelled). Then at each tracing step it is known which attributes have to be reversed in order to turn the current solution back into one examined at an earlier iteration of the search. If the remaining attributes in the RCS can be reversed by exactly one move then this move is tabu in the next iteration. For single-attribute moves, for instance, the length of an RCS must be one to enforce a tabu move. Correspondingly, in a slightly modified method REM2 all common neighbours of the current solution and of an already explored one will be forbidden. These neighbours were implicitly investigated during a former step of the procedure (due to the choice of a best non-tabu neighbour) and need not be looked at again (cf. Voß (1992)).

Obviously, the execution of REM and of REM2 represents a necessary and sufficient criterion to prevent from re-visiting known solutions. Since the computational effort of REM increases if the number of iterations increases, ideas for reducing the number of computations have been developed (cf. Glover (1990) and Dammeyer and Voß (1991a)).

For applications and (sequential) comparisons of TNM, CSM, and REM see Dammeyer and Voß (1991b) and Domschke et al. (1992).

Search Intensification and Search Diversification

A general idea for reducing the computational effort in a tabu search algorithm is that of search intensification using a so-called *short term memory*. Its basic idea is to observe the attributes of all performed moves and to eliminate those from further consideration that have not been part of any solution generated during a given number of iterations. This results in a concentration of the search where the number of neighbourhood solutions in each iteration, and consequently the computational effort, decreases. Obviously the cost of this reduction can be a loss of accuracy.

Correspondingly, a search diversification may be defined as a *long term memory* to penalize often selected assignments. Then the neighbourhood search can be led into not yet explored regions where the tabu list operation is restarted (resulting in an increased computation time). An appealing opportunity for search diversification is created by the idea of REM and REM2 resulting in REMt for $t > 2$ and integer. If at any tracing step the attributes that have to be reversed to turn the current solution back into an already explored one equal exactly t moves then it is possible to set these moves tabu for the next iteration. Note that for the case of multi-attribute moves, due to various combinations of attributes to moves, even more than t moves may be set tabu in order to avoid different paths through the search space leading to the same solution. Accordingly, search diversification is obvious.

3 Parallel Machine Models

Over the years a great variety of architectures have been proposed for parallel computing. The most widely known classification of parallel machine models (although somehow limited) is given by Flynn (1966). He distinguishes four general classes based on the idea of whether single or multiple instruction streams are executed on either one or multiple data set streams:

- **SISD** *(Single Instruction, Single Data)* including the classical sequential computers
- **SIMD** *(Single Instruction, Multiple Data)* including vector computers and array processors
- **MISD** *(Multiple Instructions, Single Data)*
- **MIMD** *(Multiple Instructions, Multiple Data)* with the processors performing each successive set of instructions either simultaneously (synchronous) or independently (asynchronous)

The above classification of parallel machine models may lead to different classes of parallel algorithms. *Vectorized algorithms* operate uniformly on vectors of data sets (SIMD). *Systolic* ones operate rhythmically on streams of data sets (SIMD and synchronous MIMD). *Parallel processing algorithms* operate on a set of synchronously communicating parallel processors (synchronous MIMD). Correspondingly, asynchronous communication leads to *distributed processing algorithms* (asynchronous MIMD and neural networks).

In addition to architectural aspects communication networks are used to classify parallel machine models. For instance, it makes a difference whether processors have simultaneous access to a shared memory, allowing communication between two arbitrary processors in constant time, or whether they communicate through a fixed interconnection network. Less formally, in certain models it is assumed that there is a *master* processor controlling the communication of the network, with the remaining processors of the network called *slaves.* For a comprehensive survey on parallel machines and algorithms see e.g. Akl (1989) and Van Leeuwen (1990).

The quality of parallel algorithms may be judged by a number of quantities, the most important one being the *speedup*, which is the running time of the best sequential implementation of the algorithm divided by the running time of the parallel implementation executed on a number of p processors. Similarly, given a prespecified time limit (cf. footnote 1) a *scaleup* may be defined as the ratio of the average problem sizes solvable with a parallel implementation to a sequential implementation of the algorithm. With heuristics, the solution quality attainable may also be measured. The *processor utilization* or *efficiency* is the speedup divided by p. The best one can achieve is a speedup of p and an efficiency equal to one.

4 Parallel Tabu Search Algorithms

Due to the success and the underlying simplicity of the main idea of tabu search, recently some implementations on parallel computers have come up tailored to specific problems.

Surprisingly, to the best of our knowledge, they are solely devoted to problems using the notion of paired-attribute moves: the travelling salesman problem, the job shop problem, and the quadratic assignment problem (compare Section 5).

In a first step we shall describe a *classification of different types of parallelism* that is applicable to most iterative search techniques (cf. Voß (1992)). Its basis is the idea of having different starting solutions or candidate solutions (so-called balls, motivated by the idea of mountains' like solution space where a ball is rolling to find a stable low altitude state) as well as a number of different strategies, e.g. based on various possibilities of the parameter setting or on the tabu list management.

- **SBSS** *(Single Ball, Single Strategy)*

  The algorithm starts from exactly one given feasible solution and performs its moves following exactly one strategy.

- **SBMS** *(Single Ball, Multiple Strategies)*

  The algorithm starts from exactly one given feasible solution by the use of different strategies where each strategy is performed on a different processor.

- **MBSS** *(Multiple Balls, Single Strategy)*

  The algorithm starts from different initial feasible solutions, each on a different processor. The same type of instruction, i.e. strategy, is performed on each processor.

- **MBMS** *(Multiple Balls, Multiple Strategies)*

  The algorithm starts from different initial feasible solutions performing different strategies.

In what follows we discuss the above ideas in more detail with special emphasis on further principles of parallelism within specific strategies. For ease of description we assume the notion of parallel or distributed processing algorithms.

SBSS

The single ball, single strategy idea is the simplest version, and obviously corresponds to the idea of classical sequential computations (cf. the SISD-model). This, however, does not restrict the possibility of parallelization.

Starting from an initial feasible solution, the best move which is not tabu must be performed. The search for this move may be done in parallel by decomposing the set of admissible moves into a number of subsets. E.g. in a master-slave architecture each (slave) processor may evaluate the best move in a specific subset. The best move of each subset is communicated to the master who picks the overall best as the transformed solution and also performs the tabu list management.

To restrict the amount of communication necessary for synchronizing the data each slave could determine the best possible move in its subset without observing any tabu list, while the tabu list in the same time is updated by the master. Then the master picks among all answers the best which is not tabu. If no such move exists, a second trial must be made while each processor has to receive and to observe the tabu list. Otherwise the next iteration is to be performed.

Additional ideas may be developed with respect to the specific strategies. In TNM, the tabu list management may be done by each processor itself by simply providing the most recent move (whose complement will be in the list). In CSM, the master builds the cancellation sequences and partitions them to the slaves, i.e., every slave has to evaluate a certain number of sequences. In subsequent iterations, the attributes of the current moves are communicated. Whenever a cancellation sequence is reduced to 1 it will be re-communicated to the master.

SBMS

In SBMS each processor executes a process which is one of the above tabu search strategies with different tabu conditions and parameters, like e.g. REMt for various t. For TNM this can be different (eventually randomly modified) tabu list lengths; for CSM, different tabu durations may be considered. The (slave) processors are halted after a prespecified time and the results are compared and the best one is calculated. A restart is possible with the best or a good seed solution. Each strategy may take a different path through the search space because of different tabu list management or parameter setting. A restart may be performed either with empty running and tabu lists or with a previously encountered list.

MBSS

The multiple balls approaches start from at most p (the number of processors available) *different initial* feasible solutions, whose calculation can vary. They may be determined either randomly or by applying different heuristics to the same problem. This may also incorporate ideas involving different diversification and intensification strategies as described above. A third possibility assumes one given feasible solution and starts with a suitable subset of its transformed (neighbourhood) solutions. (Especially with REM2 it may be assured that even in future iterations there is no overlap with the initial feasible solutions of the other processors.) The single strategy approach assumes the application of exactly one tabu search algorithm with the same parameter setting for all processors.

As with SBMS, the processes may be halted after a specific time period to coordinate their results and possibly to initiate a restart with new (hopefully) improved solutions. If the processes are performed synchronously, then the stopping may be initiated after having generated, say, m successive moves. On synchronous MIMD machines the latter approach may be especially relevant. Note that the above-mentioned possibility of parallelization within SBSS is related to a method with $m = 1$ where the best transition is evaluated.[2] With respect to MBSS, this modifies to the evaluation of the p best moves usable for a restart. For $m \geq 2$ this approach may be used as a *look ahead method*.

MBMS

The multiple balls, multiple strategies approach subsumes all previous classes, allowing search within the solution space from different starting points with different methods or parameter settings.

[2] This gives reference to incorporate different candidate list strategies. (Note the correspondance to ideas of beam search, cf. Glover (1990).)

5 Examples

In the sequel we sketch some of the ideas given in the previous sections with respect to well known combinatorial optimization problems. As mentioned above, we only found some work on problems with the idea of paired-attribute moves to perform the neighbourhood search. We start with respect to binary integer programming, exploiting single-attribute moves.

Consider the SBSS concept. Also consider n decision variables in a binary problem with no (implicit or explicit) restriction on the number of variables set to either 1 or 0. We may define simple ADD- or DROP-moves by complementing the corresponding entries of the binary variables x_i. Assume the existence of $n + 2$ processors with $n + 2$ being the master processor. The tabu list management is performed by processor $n + 1$. In any iteration of the search, each of the synchronously controlled processors $i \in \{1, \ldots, n\}$ receives the information whose variables' entry has been chosen to be exchanged as the most recent move. This move is performed together with the reversion of x_i. This usually can be done quite efficiently by reconstructing the previous solution stored at i with at most one assignment complemented. Then i offers its objective function value to the master who re-calls all results of processors referring to non-tabu moves (evaluated by processor $n + 1$). Obviously this approach may be generalized in various ways to the more general classes described above.

This concept may be applied, e.g., to the warehouse location problem (WLP), to Steiner's problem in graphs (SP), and to the multiconstraint zero-one knapsack problem (MCKP). E.g., for WLP this neighbourhood search means a reallocation of costumers, i.e., opening a new location i results in re-allocating all costumers for which i is closer than the depot currently used. Correspondingly, closing a location i forces all costumers receiving service from i to its second nearest location.

An even more challenging reoptimization problem arises within SP. There, an iteration of the neighbourhood search may consist of changing a node-oriented binary variable and calculating a minimum spanning tree (MST) on the set of all nodes with entry 1 of the corresponding variables. The question is, whether reoptimization may be carried out either by solving the modified problem anew or by starting from a previous optimal solution found by the same processor (see Glover et al. (1992) for a corresponding sequential approach with respect to MST).

If the number or weighted number of variables with value 1 is limited (as for MCKP) or fixed (as e.g. in the p-median problem) then the same approach may be applied with combined ADD/DROP- or SWAP-moves leading to paired-attribute moves.

Malek et al. (1989) follow the SBMS approach to solve travelling salesman problems (TSP) by TNM with the 2-opt exchange as moves. The tabu attributes follow different strategies in that they are restricted either to one or to the two cities that have been swapped or to the cities and their respective positions in tour. In addition different tabu parameters were used on different processors. For another parallel tabu search algorithm for the TSP see Fiechter (1990).

The quadratic assignment problem (QAP) is treated by Chakrapani and Skorin-Kapov (1991, 1992) by the use of SBSS and TNM with search intensification and search diversification performed sequentially while evaluating the moves in parallel. The set of moves is partitioned into disjoint subsets, each one on a different processor as described above.

The neighbourhood search is performed by pairwise interchanges such that for $O(n^2)$ processors available all moves can be evaluated in constant time, achieving a speedup of $O(n^2/\log n)$. Battiti and Tecchiolli (1992) use TNM together with a hashing function and compare their algorithm also with a parallel genetic algorithm. Another parallel algorithm for QAP based on TNM (with randomly varying tl-size) has been presented by Taillard (1991). It is an SBSS approach, too. The same idea has also been applied to the job shop as well as to the flow shop problem (see Taillard (1989, 1990)). The latter, in fact, also describes a single-attribute based implementation with attributes corresponding to objective function values. Chakrapani and Skorin-Kapov (1992) is especially relevant since its implementation is based on a connectionist approach related to a Boltzmann machine (cf. Aarts and Korst (1989)).

6 Conclusions

We have summarized some ideas for developing parallel tabu search algorithms. Motivated by a famous classification scheme for parallel machine models we proposed a classification scheme for parallel tabu search algorithms. While research in this field is still in its infancy we believe that reasonable achievements in the following two aspects will be provided.

- Development of a framework for a general parallel tabu search algorithm that can be applied to a wide range of combinatorial optimization problems.
- Empirical results for parallel tabu search algorithms tailored to specific problems.

Some results known from the literature (cf. Section 5) support this feeling. Despite the emphasis on parallel tabu search, sequential testing is still far from complete. In addition, the tabu search metastrategy should be tested on different classes of parallel algorithms and machine models. Especially relevant seems to be a comparison of algorithms tailored to different hardware specifications like vector computers versus synchronous and asynchronous MIMD machines. However, one should take into account identical user specifications with respect to tabu search (e.g. parameter setting, definition of the neighbourhood). Note that our classification scheme is not restriced to parallel tabu search, but may be applied for nearly any iterative search procedure, such as simulated annealing or genetic algorithms.

References

[1] E. Aarts and J. Korst (1989), *Simulated Annealing and Boltzmann Machines* (Wiley, Chichester).

[2] S.G. Akl (1989), *The Design and Analysis of Parallel Algorithms* (Prentice-Hall, Englewood Cliffs).

[3] R. Battiti and G. Tecchiolli (1992), Parallel biased search for combinatorial optimization: genetic algorithms and tabu. *Microprocessors and Microsystems*, to appear.

[4] J. Chakrapani and J. Skorin-Kapov (1991), Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, to appear.

[5] J. Chakrapani and J. Skorin-Kapov (1992), A connectionist approach to the quadratic assignment problem. *Computers & Operations Research* **19**, 287-295.

[6] F. Dammeyer, P. Forst and S. Voß (1991), On the cancellation sequence method of tabu search *ORSA Journal on Computing* **3**, 262-265.

[7] F. Dammeyer and S. Voß (1991a), Dynamic tabu list management using the reverse elimination method. *Annals of Operations Research*, to appear.

[8] F. Dammeyer and S. Voß (1991b), Application of tabu search strategies for solving multiconstraint zero-one knapsack problems. Working paper TH Darmstadt.

[9] W. Domschke, P. Forst and S. Voß (1992), Tabu search techniques for the quadratic semi-assignment problem. In: G. Fandel, T. Gulledge and A. Jones (eds.), *New Directions for Operations Research in Manufacturing* (Springer, Berlin), 389-405.

[10] C.-N. Fiechter (1990), A parallel tabu search algorithm for large traveling salesman problems. Paper presented at 1st Int. Workshop on Project Management and Scheduling, Compiegne.

[11] M.J. Flynn (1966), Very high-speed computing systems. *Proc. IEEE* **54**, 1901-1909.

[12] F. Glover (1989), Tabu search - part I. *ORSA Journal on Computing* **1**, 190-206.

[13] F. Glover (1990), Tabu search - part II. *ORSA Journal on Computing* **2**, 4-32.

[14] F. Glover, D. Klingman, R. Krishnan and R. Padman (1992), An in-depth empirical investigation of non-greedy approaches for the minimum spanning tree problem. *European Journal of Operational Research* **56**, 343-356.

[15] F. Glover and M. Laguna (1992), Tabu search. Working Paper Univ. of Colorado at Boulder, to appear.

[16] M. Malek, M. Guruswamy, M. Pandya and H. Owens (1989), Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research* **21**, 59-84.

[17] E. Taillard (1989), Parallel taboo search technique for the jobshop scheduling problem. Working Paper Ecole Polytechnique Federale de Lausanne.

[18] E. Taillard (1990), Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research* **47**, 65-74.

[19] E. Taillard (1991), Robust taboo search for the quadratic assignment problem. *Parallel Computing* **17**, 443-455.

[20] J. Van Leeuwen (1990), *Algorithms and Complexity* (Elsevier, Amsterdam).

[21] S. Voß (1992), Tabu search: Applications and prospects. Working Paper TH Darmstadt, to appear.

A Dual Strategy for Solving the Linear Programming Relaxation or a Driver Scheduling System

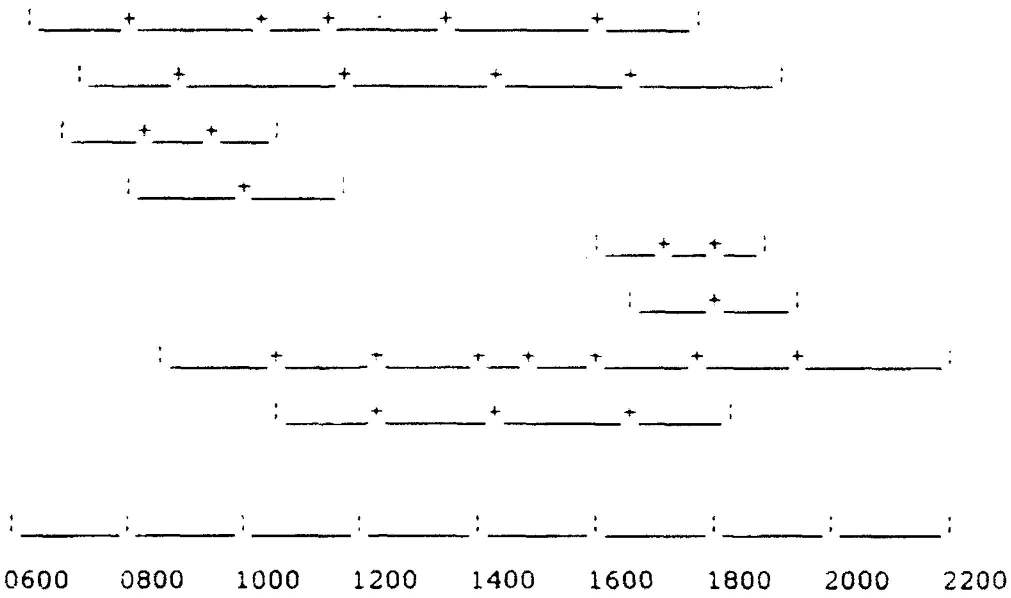
Phil Willers*, Les Proll** and Anthony Wren**

* School of Computing and Mathematics
University of Teesside
Middlesbrough
Cleveland TS1 3BA
United Kingdom

** School of Computer Studies
University of Leeds
Leeds LS2 9JT
United Kingdom

The work of a transport company (bus, train, etc.) may be represented by a *schedule* which specifies the journeys to be undertaken. Figure 1 is a graphical representation of part of such a schedule, with each line showing the times that a service begins and ends, and each '+' showing the time of a *relief opportunity* at which the driver of that service may be replaced by another driver. An indivisible period which must be worked by the same driver (e.g. between two consecutive relief opportunities) is called a *workpiece*.

Figure 1 - Graphical Representation of a Schedule



Each driver's working day consists of a number of workpieces. A complete specification of a driver's working day, including sign-on, sign-off and mealbreak times, is called a *duty*. Every transport company has many conditions that its duties must satisfy, usually called the "union agreement". This agreement may specify, for example, the maximum length of a working day and durations of mealbreaks. There is usually a very large number of different duties that could be used to cover a schedule.

There are several computer systems which can be used to determine a set of valid drivers' duties to cover a schedule provided by a transport company. This paper will consider enhancements that have recently been devised for one such system called IMPACS (Integer Mathematical Programming for Automatic Crew Scheduling). This system was developed at the University of Leeds by Wren & Smith[1] and is now marketed by the Hoskyns Group. IMPACS has mainly been used by bus companies (throughout the world) but it has also been used by train and tram companies.

At the heart of the IMPACS system is an integer Programming model which has two pre-emptively ordered objectives: to minimise the total number of duties used to cover a given schedule and to minimise a cost function which reflects both the wage cost and undesirable features of duties. The model's constraints ensure that all workpieces are covered at least once, with some specially selected workpieces being covered exactly once. Also, each duty is classified according to its type (e.g. early, late, overtime) and side constraints can be added which limit the number of duties of any type that are to be used.

Thus, the model is of the mixed set covering/partitioning type, possibly with the addition of side constraints. Ongoing research attempts to exploit further the special structure of the IMPACS model and to take advantage of recent developments in mathematical programming algorithms.

The IMPACS model has previously been solved using the following four-stage process. For the first three stages, the Linear Programming relaxation of the model is used.

Stage 1 Minimise the total number of duties using a Primal Simplex algorithm.

Stage 2 Add a constraint which ensures that the integral number of duties does not increase and minimise the cost function using a Primal Simplex algorithm.

Stage 3 If the total number of duties is not integral, add a suitable constraint, and reoptimise using a Dual Simplex algorithm.

Stage 4 Determine an integer solution using Branch and Bound techniques with constraint branching.

Optimisation within the IMPACS system is based on Ryan's ZIP package[2]. The performance of this package has been improved by incorporating Goldfarb & Reid's Primal Steepest Edge algorithm[3] and a Dual Steepest Edge algorithm due to Forrest & Goldfarb[4].

This paper will consider a new strategy for solving the Linear Programming relaxation. Enhancements to stage 4 are the subject of separate work.

Each of stages 1 and 2 of the previous strategy typically involve a large number of iterations, resulting in the time to solve the Linear Programming relaxation being a significant proportion of the total solution time. This is due to the objectives for stages 1 and 2 being different and the high degree of degeneracy inherent in the model. Also, the constraint that is added at stage 2 is fully dense, and this substantially increases iteration timings.

These difficulties have been addressed by:

1. Using a single weighted objective function.

and 2. Solving the resulting model using a Dual Steepest Edge algorithm.

The weight that is used to combine the two objectives is relatively small, and is determined by applying an algorithm due to Sherali[5] to the IMPACS model. To initiate the Dual Simplex algorithm, an heuristic has been developed to produce initial basic dual feasible solutions.

The paper will conclude with the presentation of computational results for real world problems with numbers of constraints in the range 125 to 450 and numbers of variables in the range 4000 to 11000. The results suggest that the new strategy significantly reduces solution timings.

References

1. Wren A & Smith B M (1988) Experiences with a Crew Scheduling System based on Set Covering. In Computer-Aided Transit Scheduling (Eds. Daduna J R & Wren A), pp. 160-174, Springer-Verlag, Berlin.

2. Ryan D M (1980) ZIP - A Zero-One Integer Programming Package for Scheduling. Report CSS 85, AERE, Harwell, Oxfordshire, United Kingdom.

3. Goldfarb D & Reid J K (1977) A Practicable Steepest Edge Simplex Algorithm. Mathematical Programming 4, pp. 26-29.

4. Forrest J J & Goldfarb D (1991) Steepest Edge Simplex Algorithms for Linear Programming. IBM Research Report, T J Watson Research Center, Yorktown Heights, New York 10598, USA.

5. Sherali H D (1982) Equivalent Weights for Lexicographic Multi-Objective Programs: Characterizations and Computations. EJOR 18, pp. 57-61.

Extended Abstract

Applied Mathematical Programming and Modelling Symposium
Budapest, Hungary, January 1993

DERIVING THE DUAL OF AN INTEGER PROGRAMME: ITS INTERPRETATIONS AND USES

H.P. Williams

Faculty of Mathematical Studies, University of Southampton, U.K.

This talk will begin by discussing duality in Mathematics in a wider context e.g. in the areas of Set Theory and Logic, Projective Geometry and Convex Polytopes. Some of the **mathematical** properties which are normally expected of a dual will be listed e.g. **Reflexivity** and **Symmetry**. Linear Programming (LP) and Congruence duality will then be examined for both its mathematical properties and **computational** and **economic** uses e.g. **Proving Optimality**, **Sensitivity Analysis** and **Pricing Imputation**.

A number of possible *Integer Programming* (IP) duals will be mentioned e.g. the Gomory-Baumol dual, Lagrangean dual and Surrogate dual. They all lack some of the above properties and in particular do not provide a guaranteed proof of optimality.

It will be suggested that the most satisfactory dual arises from examining the **Value Functions** and **Consistency Testers** of IPs. For Pure IPs (PIPs) these take the form of **Gomory Functions**. Gomory functions are built up by the repeated applications of the operations of

(i) Non-negative linear contributions.
(ii) Integer round-up.
(iii) Taking Maxima.

These can be expressed in the form

$$\mathrm{Max}(C_1, C_2, \ldots, C_n) \qquad (1)$$

where the C_i are **Chvátal-Functions** which are built up from operations (i) and (ii).

By comparison the Value function of the Consistency Tester of the corresponding LP relaxation will involve functions of the form

$$\mathrm{Max}(\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_r) \qquad (2)$$

where $r \leq n$ and $\bar{C}_i$ is obtained from C_i by dropping the operation (ii).

The $\bar{C}_i$ will therefore be non-negative linear conbinations of the right-hand-side coefficients, arising from the dual vertices of the LP.

It will be shown that those C_i which correspond to a $\bar{C}_i$ in (2) can be obtained by finding the Value function of PIPs over cones. This may be done by obtaining the Hermite Normal Form of the corresponding basis matrix for the LP relaxation. The resulting doubly recursive function of the right-hand-side coefficients gives the Value function (and Consistency tester). It is suggested that the depth of this recursion is a measure of complexity. The problem of extending this method to give the Value function and Consistency tester for a general PIP will be considered.

It will be shown that the Value function for a Mixed IP (MIP) is not generally a Gomory function although the Consistency tester is. By incorporating this objective as a constraint and finding the consistency tester of this system it is then possible to characterise the Value function of the MIP.

The Value function for certain MIP applications has considerable economic importance since it shows how indivisible resources should be "priced". This aspect will be considered in relation to the Fixed Charge Problem and the Power Systems Loading Problem.

1 General Problem Description

Analysts frequently face the following problem: given a multivariate (possibly correlated) population, how does one determine a good estimate of the probability function (or some number of its moments) for a complicated function of the population's variables? The primary problem to consider then is what is the most *efficient* way to sample from the input population, especially when sampling is extremely expensive and must therefore be limited to a predetermined (small) sample size. The desire is to generate a sampling plan which will be representative of the population, and produce estimates of moments which have desirable statistical properties. However, since the larger the sample, the larger the cost, there is a trade-off between generating the best estimates and reducing the amount of sampling. In order to obtain better estimates from sampling, analysts may determine them by using data collected from a stratified sampling of the population.

A special form of stratified sampling is latin hypercube sampling. In this stratification, the cumulative distribution function for each of the n population variables is divided into m blocks. The intersection of these blocks makes up a hypercube having m^n cells. If all m^n cells were sampled, the sampling approach would be a "full factorial design". Since sampling is assumed to be expensive, LHS limits the sampling to only m of the m^n possible cells. Thus, a LHS plan is not a hypercube, but is equivalent to a $m \times n$ matrix such that each of the m rows defines one sampling cell of a m^n hypercube.

The i^{th} row of a LHS sampling plan makes up what will be referred to as "run i". Defining this grouping as a *run* is motivated by the fact that typical applications of LHS involve computer-based models where the number of runs, m, is predetermined. To ensure that a plan offers a cross section of the sampling space, an additional feature of LHS is that each block of each variable must be picked once. Thus, each column of a LHS plan is a permutation of the numbers 1 to m.

Obtaining Minimum-Correlation La'in Hypercube Sampling Plans Using Discrete Optimization Techniques

Dr. Leslie-Ann Yarrow
Department of Mathematics and Statistics
Brunel University
Uxbridge, Middlesex UB10 3PH England

Combinatorial optimization, by its broad nature, has been used to model and solve a variety of problems including those arising in decision, engineering, and physical sciences. The focus of this work is to consider the solution of a sampling design problem using combinatorial optimization. The particular design problem of interest here is minimum-correlation latin hypercube sampling (hereafter referred to as MCLHS). The central point of this research is the development of combinatorial optimization procedures which provide MCLHS plans. This is an entirely new approach for finding MCLHS plans.

We introduce integer programming (IP) formulations of this problem and develop a procedure for determining minimum-correlation sampling designs. We provide the obvious IP formulation of the MCLHS problem which results in a problem having an exponential number of variables and a large (polynomial) number of constraints. We then transform the problem into a sequence of assignment problems with side knapsack equations, having a polynomial number of variables. This decomposition was found by exploiting the special structure of the problem and finding tight objective function lower bounds. We note that even after the decomposition, the problem still belongs to the NP-hard class. Although the decomposition and subsequent development of solution procedures for the smaller problems are discussed within the context of the sampling design problem, the approach may be applicable to various permutation-related IP problems such as the general quadratic assignment problem, assignment problems with side constraints, and the asymmetric travelling salesman problem variation where the objective is to find a tour which meets a specific cost value. Thus, while the research presented here focuses on solution approaches for the MCLHS problem, the general theory and findings might well prove useful for the solution of other problems known to be NP-complete.

We begin with a description of the general LHS and MCLHS problems, followed by integer programming formulations and a discussion of optimization procedures developed.

To describe the standard approach to LHS, we begin by writing the vector of variables as $(X_1, X_2, ..., X_n)$ and assume for the time being that the variables are mutually independent. The range of each X_i is divided into m (= number of runs) ascending intervals of equal probability and a random value is drawn on each interval for each variable. Next, we generate the order in which the m values of each variable are to be used in each run by creating a sequence of n random permutations of the integers 1 to m. Finally, we form the required vector for the i^{th} run by taking the i^{th} number from each of the n random permutations.

Latin hypercube sampling plans generated by the standard approach are restricted only in the sense that for each variable, a value must be picked once and only once from each of its m intervals. A point we have not yet considered is the impact that correlations between the columns of a LHS sampling plan may have on the generated estimates. For ease of explanation, we will continue the assumption that the population variables are mutually independent, although similar results are obtained for any given population covariance matrix. For the n variables, although their sampling plan permutations are determined independently, a standard LHS plan will, in general, have some level of correlation between the pairs of permutations. Thus, the sampling plans will not, in general, parallel the correlations of the true joint distributions. If LHS sampling is done without concern for the correlation pattern (or lack thereof), the estimators cannot be guaranteed to be unbiased or even consistent.

The desire then is to design LHS plans which incorporate the variables' true pairwise correlations. For two variables, X_i and X_j, with distribution functions having strictly positive standard deviations, σ_i and σ_j, the correlation coefficient between the variables is defined as

$$\rho_{ij} = \frac{\text{cov}(X_i, X_j)}{\sigma_i \sigma_j}$$

where $\text{cov}(X_i, X_j)$ denotes the covariance between variables X_i and X_j.

To *approximate* the pairwise correlation coefficients ρ_{ij}, we will consider the correlation coefficients between the pairs of LHS plan permutations associated with variables X_i and X_j. (The two forms of correlations are equal when X_i and X_j are both uniformly distributed.) For permutations of the integers from 1 through m, it can be shown that the correlation coefficient of the indices of any pair of permutations is

$$\hat{r} = 1 - \frac{6\sum_{v=1}^{m} D_v^2}{m(m^2 - 1)},$$

where D_v is the difference between the v^{th} integer elements in the vectors. This is known as the Spearman rank correlation coefficient and can take on values in the interval $[-1, 1]$. The expected value of the rank correlation coefficient is 0, and its variance is $1/(m - 1)$. Throughout the remainder of this paper, we denote the rank correlation estimate between the column permutations of variables X_i and X_j by $\hat{r}_{ij}$.

For illustration, suppose we want to run a model with three mutually independent uniformly distributed variables (for simplicity, x, y, and z), each to be represented by values chosen from their respective sample spaces. Assuming further that we are allowed only six runs, consider the LHS plan given below:

Table 1: Latin Hypercube Sampling Example

Model Run	Variable Values		
1	x_1	y_1	z_5
2	x_2	y_6	z_3
3	x_3	y_5	z_4
4	x_4	y_3	z_1
5	x_5	y_2	z_2
6	x_6	y_4	z_6

The rank correlation coefficients for this example are

$$\hat{r}_{12} = 0.00, \;\; \hat{r}_{23} = 0.00, \;\; \hat{r}_{13} = 0.00,$$

and hence, it appropriately models the mutual independence of the three variables. If, for example, the variables were dependent with true joint distributions having pairwise rank correlations of say, $r_{12} = -.6$, $r_{23} = -.42$, and $r_{13} = .14$, then this particular sampling plan would not suitably parallel these true rank correlations.

The objective of the restricted LHS problem we consider is the selection of column permutations which attempt to meet exactly the true rank correlations associated with the variables. In this way, sampling is intended to match more closely the true marginal distributions of the input variables. Specifically, the minimization problem, called minimum-correlation LHS (MCLHS), provides a sampling specification minimizing the sum of the absolute values of the pairwise differences $(\hat{r}_{ij} - r_{ij})$. In much of the discussion however we will *minimize* the sum of the absolute values of the pairwise rank correlations $\hat{r}_{ij}$. This models the situation when independence of the variables is likely (i.e., $r_{ij} = 0$).

2 Integer Programming Models for MCLHS

The minimum-correlation latin hypercube sampling problem described can be formulated as a n-index assignment problem with side knapsack equation constraints (APSEC). To begin, define:

$$x_{v_1 \ldots v_n} = \begin{cases} 1 & \text{if } v_1 v_2 \ldots v_n \text{ is a sampled cell} \\ & \text{where the } n\text{-indices on the } x\text{-variable,} \\ & \; v_1, v_2, \ldots, v_n, \text{ can each take a value from 1 to } m \\ 0 & \text{otherwise} \end{cases}$$

and also d_{ij}^{+}, $d_{ij}^{-} \in \mathcal{R}_1^{+}$ such that:

$$d_{ij}^{+} - d_{ij}^{-} = (\hat{r}_{ij} - r_{ij})\, m(m^2 - 1)/6 \qquad i = 1 \ldots n, \;\; j > i.$$

Thus, d_{ij}^{+} and d_{ij}^{-} are the positive and negative magnitudes of the deviations from the true rank correlation of the rank correlation between column permuations i and j.

Equivalent to minimizing the sum $\sum_{j=2}^{n} \sum_{i=1}^{j-1} \frac{m(m^2-1)}{6} \mid \hat{r}_{ij} - r_{ij} \mid$, is minimizing the objective function

$$\min \left\{ \sum_{i=1}^{n-1} \sum_{j>i} \;\; (d_{ij}^{+} + d_{ij}^{-}) \right\}.$$

Although the formulations described below are applicable to cases with nonzero r_{ij}, for ease of presentation, we will assume $r_{ij} = 0$. It can be shown that

$$m(m^2 - 1)\hat{r}/6 \; = \; m(m^2 - 1)/6 \; - \; \sum D_v^2$$

will be integer-valued for all pairs of permutations. Thus, we can now define d_{ij}^{+}, $d_{ij}^{-} \; \in \mathcal{Z}_1^{+}$ such that:

$$d_{ij}^{+} - d_{ij}^{-} = m(m^2 - 1)\hat{r}_{ij}/6 \qquad i = 1 \ldots n, \;\; j > i.$$

In order that the IP formulation fully encompasses the MCLHS, it must include assignment constraints that draw a one-to-one correspondence between the positive-valued $x_{v_1 \ldots v_n}$ of a feasible solution and n-tuples of column permutations. Thus, the j^{th} column permutation requires the m assignment constraints

$$\underbrace{\sum_{v_1=1}^{m} \sum_{v_2=1}^{m} \cdots \sum_{v_n=1}^{m}}_{\text{excluding } \sum_{v_j=1}^{m}} x_{v_1 \ldots v_n} \; = \; 1 \qquad v_j \; = \; 1 \ldots m$$

Additional constraints are needed to enforce that
$m(m^2 - 1)\hat{r}_{ij}/6 = d_{ij}^{+} - d_{ij}^{-}$ holds for all i and j, $i < j \leq n$. These constraints are

$$\sum_{v_1=1}^{m} \sum_{v_2=1}^{m} \cdots \sum_{v_n=1}^{m} (v_i - v_j)^2 \, x_{v_1 \ldots v_n} \; = \; m(m^2 - 1)/6 \quad \forall \, i < j \leq n$$

In addition to belonging to the class of NP-complete problems, we see that this formulation requires m^n x-variables as well as a total of $n(n-1)$ deviational (d_{ij}^{+}, d_{ij}^{-}) variables. There are nm assignment constraints and $\binom{n}{2}$ constraints to ensure that $m(m^2 - 1)\hat{r}_{ij}/6 = d_{ij}^{+} - d_{ij}^{-}$. Hence, although this formulation is the most straightforward, we will present other APSEC formulations which have more reasonable problem size growth.

To develop alternative formulations, we use the objective function lower bound of $\binom{n}{2}6/(m(m^2-1))$ when $m = 2 + 4l$ for some nonnegative integer l, and zero otherwise, and make the assumption that given $k-1$ column permutations with minimum $\sum_{j=2}^{k-1}\sum_{i=1}^{j-1} \mid \hat{r}_{ij} \mid$, it is possible to fix these columns and find an optimal k^{th} column permutation. Our research and empirical results have shown that these are valid assumptions.

Suppose we have a solution to the $(k-1)$-dimensional problem, and wish to use this solution to obtain a solution to the k-dimensional problem. Let $(p^1, p^2, \ldots, p^k)$ denote the corresponding column permutation vectors, and define

$$x_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ element of column } k \\ & \text{is assigned value } j \\ 0 & \text{otherwise} \end{cases}$$

To ensure that column k is a permutation of numbers $1 \ldots m$, we add the assignment constraints :

$$\begin{aligned} \sum_i x_{ij} &= 1 \qquad\qquad j = 1, \ldots, m \\ \sum_j x_{ij} &= 1 \qquad\qquad i = 1, \ldots, m. \end{aligned}$$

We see that the positive elements of an x-solution define a k^{th} column. We will henceforth interchangably use the terms "an x-solution" and "the k^{th} column defined by the positive elements of x".

There are $(k-1)$ additional constraints of the following form:

$$(1) \quad d_{tk}^{+} - d_{tk}^{-} = m(m^2-1)/6 - \textstyle\sum_{i=1}^{m}\sum_{j=1}^{m}(p_i^t - j)^2 x_{ij} \qquad t = 1, \ldots, k-1$$

where p_i^t is the i^{th} entry of the column permutation vector p^t. With these constraints, we implicitly fix the $(k-1)$ previourly found column permutations.

The formulation defined thus far with objective function

$$\min \sum_{i=1}^{k-1} (d_{ik}^+ + d_{ik}^-),$$

is a general formulation for finding a k^{th} column permutation, having fixed the $(k-1)$ column permutations that minimize $\sum_{i=1}^{k-2} \sum_{j>i} \mid \hat{r}_{ij} \mid$. Empirical evidence strongly supports that there exists a k^{th} column that meets the lower bound for $\mid r_{ik} \mid$, $i = 1, \ldots, k-1$. Hence, there will exist a solution to the assignment constraints that generates a k^{th} column satisfying

$\mid m(m^2-1)\hat{r}_{ik}/6 \mid =$

$$\mid m(m^2-1)/6 - \sum D_v^2 \mid \; = \begin{cases} 1 & \text{if} \quad m = 2+4l, \;\; l \in Z_1^+ \\ 0 & \text{otherwise} \end{cases}$$

for all $i = 1, \ldots k-1$. To incorporate this into the formulation, we require that d_{tk}^+ and d_{tk}^-, $i = 1, \ldots k-1$ be binary variables. For
$m \neq 6+4l$, $l \in Z_1^+$, any solution that obtains the lower bound must have $d_{tk}^+ + d_{tk}^- = 0$. If however, $m = 2+4l$, $l \in Z_1^+$, we can conclude that $d_{tk}^+ + d_{tk}^- = 1$, $\quad t = 1, \ldots k-1$. In either case, the problem can be restated as a *feasibility* problem with no objective function. We shall refer to this feasibility assignment problem with side equations as **FASE**.

The **FASE** formulation follows the conjecture that one can iteratively solve k-dimensional problems using the previously determined $(k-1)$-dimensional solutions. Thus, rather than solving one large APSEC program with $m^n + n(n-1)$ variables and $nm + \binom{n}{2}$ constraints, one could solve a sequence of smaller two-dimensional **FASE** problems with at most $m^2 + 2k$ variables and $2m + k$ constraints ($2 \leq k < n$).

In the presentation, we shall discuss heuristic and Lagrangean-based solution procedures developed to solve the MCLHS problem and its equivalent formulation FASE.

Late Papers

Introduction to Object-Oriented Modeling with GAMS 2.25

Robert Entriken

December 7, 1992

Abstract

This paper describes a standard for the use of GAMS 2.25 as an object-oriented modeling language. The over-riding benefit of using this technique is the ease with which many individuals can simultaneously develop extraordinarily complex modeling systems. Lesser, but still important benefits include: structured user-interface design, plug-in/plug-out models, isolating portions of the problem, easy maintenance and updates, and model re-use. Simultaneous model development stems from the latter benefits, while all of these advantages derive from the clear, rigorous organization of your model as specified in the following standard.

We present the concepts of encapsulation (forming objects) and hierarchical modeling in the context of mathematical modeling. Encapsulation is a well-known programming technique that is newly applied to modeling, and our version of hierarchical modeling differs slightly from past notions. Traditionally, a hierarchical model embodies the concept of forming larger models from a collection of sub-models. The following method is based on a partition of the relations (equations) of the model, where the elements of the partition are partially ordered.

1 Overview

Object-oriented modeling (OOM) is a method of modeling that closely imitates object-oriented programming (OOP) [?,?,?]. We have developed a standard for using GAMS 2.25 [?] as an OOM language. The difference being that the OO models are much more structured and abstract. This makes them more user friendly because their use is well defined by the structure and their details are hidden within. OO Models thus appear simpler and more uniform to the user.

Four essential properties set OOM apart from standard GAMS 2.25:

Routines: Structuring the assignment statements into procedures as in Pascal.

Encapsulation: Combining data and variables with the equations and assignment statements that manipulate them to form a new data type—a model.

Information Inheritance: Defining a model that uses other models in its formulation, with each sub-model inheriting the information from its ancestors. The use of models within models defines the *use hierarchy* which forms a partial ordering of all used models.

Polymorphism: Giving a model's routine one name that is shared by all descendants in the use hierarchy, with each descendant implementing the routine in a way appropriate to itself.

Routines are implemented using the `$INCLUDE` statment. Encapsulation, inheritance, and polymorphism are implemented in GAMS 2.25 through self discipline. The following is a detailed discussion of the principles and implementation of OOM in GAMS 2.25 through self-discipline. We hope that the future will bring the language extensions need for a proper implementation. In which case, the standard described below would be enforced by the compiler.

There are now a variety of experimental modeling languages offering object-oriented features, notably ASCEND [?] and MODEL.LA [?]. We offer a form of inheritance that differs from the class inheritance of standard OOP and OOM languages. This is an extra restriction placed on our models based on deferred requirements, and the use of models within other models. Data and variables are legated (passed down) to the descendants, while methods are used by ancestors to ensure that deferred information[1] is properly defined.

[1] Data and variables that have been declared but are yet undefined.

There is a restricted form of communication control between the models of the use hierarchy. Essentially, descendants can inspect ancestor information but ancestors can only ask that certain information be provided. In this way, siblings communicate through the parent, and its deferred information.

We further expound on these concepts and offer a full accounting of the presentation. First we introduce a model and how it is encapsulated. This leads to an overview of traditional hierarchical modeling. Then we explain how OOM fits into this background. The final section gives the standard itself—how to implement OOM in GAMS 2.25.

References

[1] Bjarne Stroustrup (1986). *The C++ Programming Language*, Addison-Wesley, Reading, Massechusetts, USA.

[2] Borland International (1990). Turbo Pascal 6.0 - User's Guide, Scotts Green, California, USA.

[3] Bertrand Meyer (1988). *Object-Oriented Software Construction*, Prentice Hall International, Hertfordshire, UK.

[4] Anthony Brooke, David Kendrick and Alexander Meeraus (1992). *GAMS - A user's Guide, Release 2.25*, The Scientific Press, South San Francisco, California.

[5] P. Piela, T.G. Epperly, K.M. Westerberg (1991). ASCEND: An object-oriented computer envieronment for modeling and analysis: The modeling language. *Computers and Chemical Engineering* 15, 53–72.

[6] G. Stephanopoulos, G. Henning, and H. Leone (1990). MODEL.LA.: A modeling language for process engineering. Part I. The formal framework. Part II. Multi-facetted modeling of process systems. *Computers and Chemical Engineering* 14, 813–869.

TOWARD A FORMAL THEORY TO A MODELS' INTEGRATION

M. GAGLIARDI, C. SPERA
Department of Quantitative Methods
University of Siena - Italy

e-mail: Spera@Sivax.cineca.it

EXTENDED ABSTRACT

1. Motivations for a formal theory.

The definition of a specific model is often conceived as a work which has to be done from scratch. In fact, the variety of the variables describing the modelled reality seems to exclude the possibility that a model can be defined assembling pieces of correlated sub-models. To define models from scratch greatly decreases the productivity of the work.

It seems that the keyword in increasing modelling productivity is "reusability". Models can be reused and integrated so to produce new models. Naturally models to be integrated have to be expressed using a common base and the result has to lie on the same framework. In this paper the chosen framework is the Structured Modeling, as formally defined by Geoffrion, [3].

Here we define three integration levels, according to the degree of influence of the operator in the procedures used to merge the models:

Level 1 - All the procedures are automated. This means that the user selects the input models and the genera to be integrated, and the output integrated model is automatically produced.

Level 2 - The user selects the input models and the order of integration among the genera, and the output integrated model is automatically produced.

Level 3 - The user select the input models, the genera to be integrated and formulate the steps necessary to integrate. The output integrated model is not automatically produced, since the steps can vary according to the situation.

2. Preliminary results.

In the rest of this paper we assume that the reader is familiar with the formal theory of the Structured Modeling.

Given a Structured Model M_i, let $G_i = \{g_j, j = 1, \dots, k\}$ be the set of all the genera: this can be partitioned into three disjoined sets: PC, A and FT such that:

PC $= \{g_j \in G_i$: g_j is a primitive or a compound entity genus$\}$
A $= \{g_j \in G_i$: g_j is an attribute genus$\}$
FT $= \{g_j \in G_i$: g_j is a function or a test genus$\}$.

Lemma 1: *Any genus $g_j \in PC_i$ does not have references to any other genera $g_k \in (A \vee FT)$*

Proof: Primitive entity elements, by definition, have no calling sequence, therefore they do not have references to any other element; compound entity elements, by definition, are construct only on primitive entity elements. ■

Lemma 2: *Any genus $g_j \in A_i$ has only references to another genera $g_k \in PC_i$.*

Proof: Attribute elements, by definition, characterize only primitive and compound elements. ■

Lemma 3: *Any genus $g_i \in FT_i$ does not have references to any other genera $g_k \in PC_i$.*

Proof: Function and test elements call, by definition, attribute, function and test elements; therefore, they cannot call primitive and compound entity elements.■

Definition 1: Connected Module, Sub-Model.
*A module is a **Connected Module** if its genera and their calling sequences define a connected graph. A **Sub-model** is a connected module with at least one primitive entity genus.*

Definition 2: Behaviour Equivalence on $\overline{FT}_i \subseteq FT_i$.
*Two structured models M_1 and M_2 are **Behaviour Equivalent** on $\overline{FT}_1 \subseteq FT_i$ and $\overline{FT}_2 \subseteq FT_2$ if the following two conditions hold:*
a) The set $\overline{A}_1$ of the attribute genera directly or indirectly called by the $g_j \in \overline{FT}_1$, and the set $\overline{A}_2$ of the attribute genera directly or indirectly called by the $g_j \in \overline{FT}_2$ have the same structure;
b) $\overline{FT}_1$ and $\overline{FT}_2$ give as output the same values.

We shortly write "behaviour equivalent" when the sub-set $\overline{FT}_i$ coincides with FT_i.

Definition 3: Normal Model.
*A model is called **normal** if an isomorphic relation exists between attribute and compound genera, and their elements.*

The graph of the elements of a normal model is shown in figure 1: dotted rectangles identify genera.

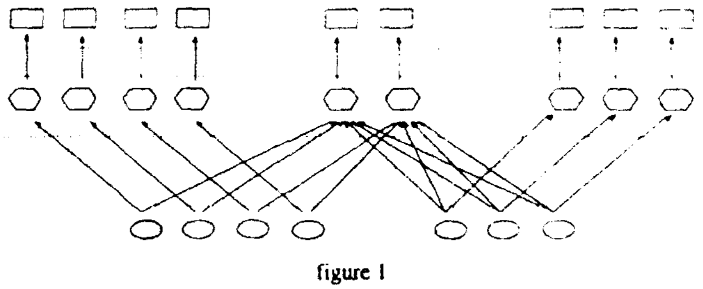


figure 1

Proposition 1. *Given a Structured Model M_i, it is always possible to construct a normal model $N(M_i)$ which is behaviour equivalent to M_i.*

Proof: Let us consider a generic attribute genus $g_j \in A_i \subset M_i$. It is always possible to define a new compound entity genus, $c_k \in PC_i$, with the same calling sequence of g_j. Lemmas 1 and 2 ensure that genera which are called by an attribute genus can be called by a compound entity genus too. An

isomorphic relation can be set among the elements of g_j and c_k: the first element of g_j calls the first element of c_k, etc. This process is repeated for every attribute genus of M_i.
If we indicate with $N(M_i)$ the modified model, the set $B = \{c_k, g_j\} \subset N(M_i) \Leftrightarrow g_j \in M_i$ for every genus $g_k \in FT_i \subset N(M_i)$. ∎

Definition 4: Index Basis, Index Basis Set.

*An Index Basis of a normal model $N(M_i)$ is a couple of genera $B_j = \{a_j, c_j\}$, where $a_j \in A_i \subset M_i$ is an attribute genus, and c_j is the compound entity genus called by a_j. The genus a_j is called **value component** of B_j, while the genus c_j is called **index component**. The set $BS_i = \{B_j, j:1, \ldots, k\}$ containing all the index basis of $N(M_i)$ is called Index Basis Set.*

Definition 5: Index Function.

*An **Index Function** $i(g_j)$ is a rule which associate to every genus $g_j \in N(M_i)$ the cardinality of its index set.*

As example, given a genus g_i indexed by $j \times k \times l$, its index function $i(g_i)$ returns as value 3.

3. Main results.

In this caragraph we try to give an example for each level of integration previously defined.

3.1. Level 1 integration example.

To show the first level of integration we need to introduce the definition of a function sub-model.

Definition 6: Function Sub-Model.

*SubM(f) is called **Function Sub-model** if the following properties hold:*

a) SubM(f) is a normal model.

b) SubM(f) has at least a function genus $f \in FT_i \subset M_i$ indexed as singleton.

In the following we give a procedure, which transforms a Structured Model M_i with at least one function genus indexed as singleton into a function sub-model.

The following procedure, CREATE_FUNCTION_SUBMODEL, needs as input a model M_i and a singleton genus $f \in FT_i \subset M_i$, and produce as output a function sub-model. The proof of this is given in Proposition 2.

```
procedure CREATE_FUNCTION_SUBMODEL (input: $M_i$, f; output: SubM(f));
/* Modify $M_i$ into a function sub-model SubM(f) */
begin
     /* step I. "Normalize the model" */
     NORMAL ($M_i$);
     /* step II. "Merge functions" */
     Create a LIST of calling sequence segments $s_i$ of f;
     repeat
          Examine the segment $s_i \in$ LIST;
          if the referred genus $g_k \in FT_i$
          then
               /* a */    Substitute $s_i$ with the calling sequence of $g_k$ ;
               /* b */    Substitute the value field of $g_k$ with its rule;
               /* c */    Delete $g_k$;
               Delete the segment $s_i$;
     until (end of LIST);
```

```
    /* step III 'Delete genera having no influence on f' */
    Create a LIST of $g_j \in M_i$;
    repeat
        Examine $g_j \in$ LIST;
        if ($g_j \in FT_i$ and $g_j \neq f$)
        then delete $g_j$;
        if ($g_j \in A_i \cup PC_i$ and $g_j$ is not called directly or indirectly by f)
        then delete $g_j$;
    until (there are no more $g_j \in FT_i \subset M_i$, $g_j \neq f$) and (there are no more $g_j \in$
    $A_i \cup PC_i \subset M_i$ not called directly or indirectly by f)
end
```

Proposition 2: *Given a Structured Model M_i and an arbitrary singleton function genus $f\in FT_i \subset M_i$, it exists a transformation T such that:*

$T(M) = SubM(f)$

and $SubM(f)$ and M_i are behaviour equivalent on f.

Proof: By applying procedure CREATE_FUNCTION_SUBMODEL which defines the procedure T.∎

Let us show how a function genus f can be reused as an input parameter for other models. This action is totally automated, here is an example.

Suppose we have two models M_1 and M_2, we want to substitute the genus $g_j \in A_1 \subset M_1$ with the computed value given by the genus $f \in FT_2 \subset M_2$. This goal is achieved applying the following procedure (the symbol $[M_1, SubM_2]$ means the integrated output model):

```
procedure REUSE (input: $M_1$, $M_2$, $g_j$, f; output: $[M_1, SubM_2]$);
/* Integrate $M_1$ and $M_2$, f is substituted to $g_j$ */
begin
    /* Step I "Changes in $M_2$" */
    CREATE_FUNCTION_SUBMODEL ($M_2$,f; $SubM_2(f)$);
    Create a LIST of genera $g_i \in A_2 \subset SubM_2(f)$;
    repeat
        Add the calling sequence's segments of $g_j \in A_1$ to the calling sequence of
        $g_i \in A_2 \subset SubM_2(f)$;
    until end of LIST;
    /* Step II "Changes in $M_1$" */
    Create a LIST of genera $g_i \in FT \subset N(M)$;
    repeat
        Select $g_i$ from LIST;
        if $g_i$ calls $g_j \in A_1$
        then
            Substitute $g_j$ with f in the calling sequence of $g_i$;
        LIST := LIST - $g_i$;
    until end of LIST
    /* Step III "Delete attribute genus" */
    Delete $g_j \in A_1$;
end.
```

Proposition 3: *Given two Structured Models M_1 and M_2, it is always possible to substitute an attribute genus $g_j \in A_1 \subset M_1$ with a singleton function genus $f \in FT_2 \subset M_2$. The result is a Structured Model.*

Proof: By applying the procedure REUSE we obtain as result the model $[M_1, SubM_2]$. Its graph of genera must be finite, closed and acyclic.

a) **Finiteness**. Step III guaranties that the number of genera of $[M_1, SubM_2]$ is equal to the number of genera of $(M_1 \cup SubM_2(f)) - g_j$.

b) **Closure**. By steps I and II, there is at least one genus of M_1 calling a genus of $SubM_2(f)$ and at least one genus of $SubM_2(f)$ calling a genus of M_1. From closure of M_1 e $SubM_2(f)$ it follows the closure of $[M_1, SubM_2]$.

c) **Acyclicity**. Let us consider an arbitrary sub-set of genera $G_i \subseteq [M_1, SubM_2]$, and let us assume that it is cyclic. Therefore, G_i contains genera belonging to both models, because no new references are set by the procedure among genera belonging to only one model. By construction, the sequence must be of the type:

$\{ \ldots, a_i \in A_2 \subset SubM_2(f), f, \ldots \}$.

The genus following f in the sequence has to be a function genus, while the genus preceding a_i has to be a compound entity genus. By Lemma 3 there are no references among function genera, and compound entity genera. Therefore, G_i cannot by cyclic. ■

Figure 2 shows how two models are integrated.

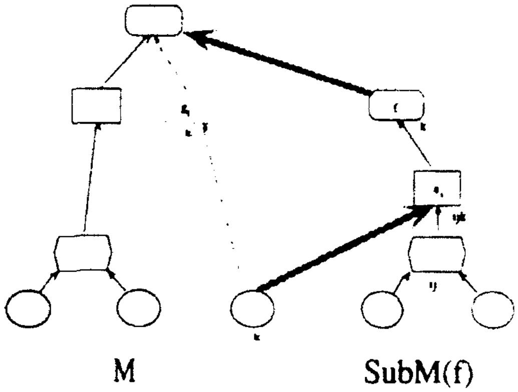


Figure 2

Proposition 4. *Given two normal models M_1 and M_2, the integrated model obtained substituting an input parameter $g_1 \in A_1 \subset M_1$, with an output parameter $g_2 \in FT_2 \subset M_2$ is a Structured Model if $i(g_1) = i(g_2)$.*

Proof: It follows the same line of proposition 3. The necessary condition given by the equality of the index functions ensures the closure and acyclicity of the graph of the elements.■

Given the result of proposition 4 the following procedure can be constructed. The input parameters are the two normal models, an index basis of the model M_1 and a function genus of the model M_2.

```
procedure USE (Input: $N(M_1)$, $N(M_2)$, $B_1$, f; Output: $\{N(M_1), N(M_2)\}$);
begin
    Select $a_1 \in B_1$;
    Compute $i_1(a_1)$;
    Compute $i_2(f)$;
    if $i_1(a_1) \neq i_2(f)$ then exit;
    Create a LIST of genera $g_i \in FT_1 \subset N(M_1)$;
    repeat
        Select $g_i$ from LIST;
        if $g_i$ has a reference to $a_1$ then
            Substitute the reference to $a_i$ with a reference to f;
        LIST := LIST - $g_i$;
    until end of LIST;
    Delete $B_1$;
end.
```

The following steps create an integrated model, which is the same result as in Geoffrion. The final graph of genera obtained applying sequentially step 0 - step IV is shown in figure 4.

```
Step  0.
NORMAL (fin);
NORMAL (mkt);
NORMAL (mar);
NORMAL (mfg);

Step  I.
SUBSTITUTE (mkt, mar, [P,D1], [P,D2]);
USE (mar, mkt, [V,D3], V);

Step  II.
USE (mfg, mar, [V,D5], V);
USE (mar, mfg, [E,D4], E);

Step  III.
SUBSTITUTE (fin, mar, [P,D6], [P,D2]);
USE (fin, mar, [E,D8], E);
USE (fin, mar, [V,D9], V);

Step  IV.
MERGE (mfg, mar, P, U);
```

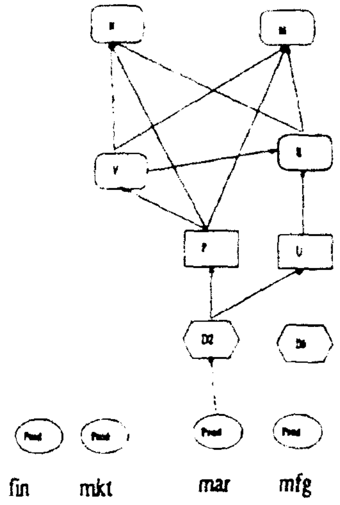


Figure 4

3.3 Level 3 integration example.

At this level of integration the user needs to define the steps to integrate the models, and there are no automated procedure. Let us present another example extracted from Geoffrion [4]. The steps are informally defined, since the user will formalize them.

```
Step  1
Delete DEM and T:DEM genera from TRANS1
Delete SUP and T:SUP genera from TRANS2

Step  II
Merge genus CUST from TRANS1 with genus
PLANT from TRANS2;

Step  III
Create new genera T:DC and define its
reference;

Step  IV  (Optional)
Create a new genus TOTS being the sum of
the TOTS function genera of the two
models;

Step  V  (Optional)
Rename genera;
```

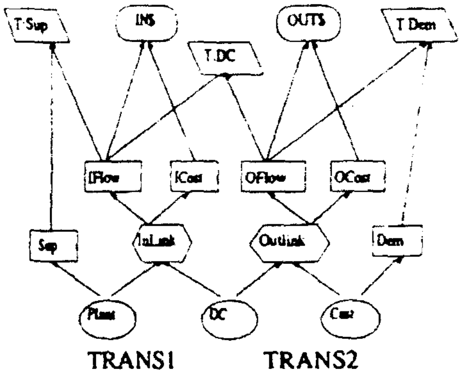


Figure 5

4. Conclusions.

The first remark about the definition of a formal theory to models' integration is *modularity*. This can be easily achieved projecting the theory of the Structured Modeling into the same space of the Object Orientation Principles.

The second remark regards the construction of three sub-sets which contain the procedures characterizing the formal rules of the three integration levels.

Both aspects will be deeper developed in the future.

3.2. Level 2 integration example.

In this case the role of the user is relevant, since the input parameters to be merged are only identified by him.

Proposition 5. *Given two normal models $N(M_1)$ and $N(M_2)$ and the corresponding index basis set BS_1 and BS_2, the integrated model obtained substituting in $N(M_1)$, $B_j \in BS_1$ with $B_k \in BS_2$ is a Structured Model.*

Proof: To substitute B_j with B_k implies that every genus $g_j \in FT_1$ has to replace the reference in its calling sequence to $a_j \in B_j$ whit $a_k \in B_k$. The graph of genera of the integrated model has to be: (a) finite; (b) closed and (c) acyclic. (a), (b) hold by construction; (c) holds by lemma 2.∎

Proposition 6. *Given two normal models $N(M_1)$ and $N(M_2)$, and the corresponding index basis set BS_1 and BS_2, the integrated model obtained substituting, in $N(M_1)$, an index component $c_j \in B_j \in BS_1$ with an index component $c_k \in B_k \in BS_2$ is a Structured Model*

Proof: It follows the lines of proposition 5.∎

Given the results of the proposition 5 and 6 the following procedures can be constructed.

```
procedure SUBSTITUTE (Input: $N(M_1)$, $N(M_2)$, $B_1$, $B_2$;
                            Output: [$N(M_1)$, $N(M_2)$]);
begin
    Create a LIST of genera $g_i \in FT_1 \subset M_1$;
    repeat
        Select $g_i$ from LIST
        Substitute $A_1 \in B_1$ with $A_2 \in B_2$ in the calling sequence of $g_i$;
        LIST := LIST - $g_i$;
    until end of LIST;
    Delete $B_1$;
end.

procedure MERGE  (Input: $N(M_1)$, $N(M_2)$, $B_1$, $B_2$;
                    Output: [$N(M_1)$, $N(M_2)$]);
begin
    Select $c_1$, $a_1 \in B_1$, $c_2 \in B_2$;
    Substitute $c_1$ with $c_2$ in the calling sequence of $a_1$;
end.
```

In the next we treat the core example extracted from Geoffrion [4]. The sub-models to be integrated are shown in figure 3 (the details are omitted):

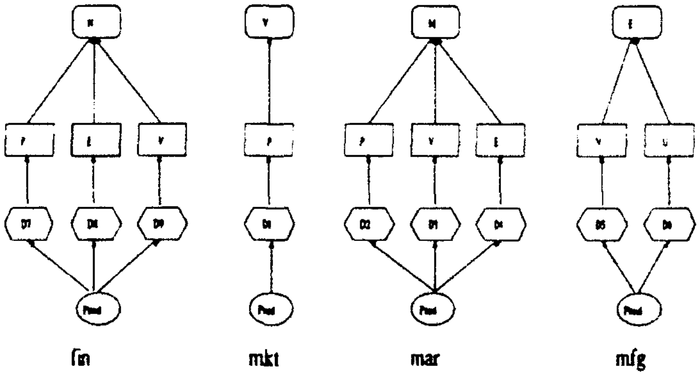


Figure 3

XI

References

[1] ANDRONICO. A., COSSA. L. , GAGLIARDI. M. e SPERA. C. (1992b)."An Object Oriented Approach to a Model Management System: Characteristics and Examples", in *Atti 36° Convegno Nazionale ANIPLA*, Bottaro, Zoppoli Editors, Pirella Genoa.

[2] GEOFFRION. A.M. (1987). "An introduction to Structured Modeling", *Management Science*, vol. 33, pp. 547-589.

[3] GEOFFRION. A.M. (1989). "The formal aspect of Structured Modeling", *Operations Research*, vol. 37, pp. 30-51.

[4] GEOFFRION. A.M. (1990c). "Reusing Structured Models via Model Integration", Working paper #362, Western Management Science Institute, UCLA.

[5] KOTTEMAN. J.E. e DOLK. D.R. (1990). "Model Integration and Modeling Languages", Working paper, Naval Postgraduate school.

[6] TSAI. Y. (1987). "An Operational Approach to Model Integration Using a Structured Modeling Framework", Research Paper, Anderson Graduate School of Management, UCLA.

The Model Recognition Phase in a Model Management System

P. LOMAGISTRO, C. SPERA
Department of Quantitative Methods
University of Siena, Italy

EXTENDED ABSTRACT

1. Introduction

As pointed out by many authors, a Model Management System (MMS) provides for creation, storage, manipulation, and access to models. MMS functions can be divided in two main groups: Model storage functions and Model manipulation functions. The former includes Model Building, Model Representation, physical and logical Model Storage and Model Retrieval; the latter includes Model Instantiation, Interface with Databases, Model Maintenance, Links between model and Algorithms, and Model Solving.

Model representation schemes plays a key role in the implementation of effective MMSs. To fully implement the functions of MMSs, we need to state a rigorous conceptual framework with a single model representation leading to:

1) independence of model representation and model solution,
2) representational independence of general model structure and detailed data needed to describe specific model instances.

A system based on these ideas would show its usefulness for most phases of the life-cycle associated with model-based work (Geoffrion 1987). For example, consider a mathematical programming problem. Once a model of this problem has been constructed, a MMS should allow the user to perform the following steps:

1) select the solution technique (if any),
2) solve the model,
3) conduce sensitivity analysis.

To automate steps 1 and 2, the system has to be able:

a) to *recognize* what kind of model arises (so that it could automatically select the appropriate solver);
b) to *translate* data instantiating the model (querying the Database where they are stored) into the format required by the selected solver.

This paper will focus on the model recognition phase. We will try to give its theoretical foundations and to define which conditions a model definition language has to satisfy so that the resulting representation is "recognizable".

Our formalization of the recognition process is based on the concept of "minimal representation". A representation of a model is minimal if any other equivalent representation of the same model can be "reduced" to it.

2. Model Recognition Problem: Preliminary Results.

The aim of this section is to provide for some formal definitions. In the next, we will use them to illustrate how recognition process can be carried out.

The recognition process we are trying to formalize is based on the concept of minimal representation. A representation of a model is minimal if any other equivalent representation of the same model can be reduced to it.

In the rest of this paper we will define and explain minimality, equivalence and reduction of model representations; first we need to define what we intend for "model" and "model representation".

Definition 1

We define the system M to be a **model** of the system P if:

— M does not interact neither directly nor indirectly with P

— M is used to obtain information about P

— M comprises all the elements of P relevant for the intended purpose of the model.

Definition 2

Given a formal language L and a model M_i, we define $L(M_i)$ to be its formal **representation** under L, if it comprises the expression in language L of all the elements of M_i and of the interactions existing among them.

In the following we will use the terms "model representation" or simply "representation" to indicate the "formal" representation of a given model under some formal language.

Let us consider, as an example, model M_i as the model of the system P that computes the mean of a given series of values belonging to P; if L is the standard algebraic notation, then $L(M_i)$ will be:

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad [1]$$

If $L(M_i)$ exists and is unique, then the recognition problem has a trivial solution, because there is a 1:1 correspondence between model and its representation. Unfortunately, except for very few cases, the model M_i has many representations $L_j(M_i)$, $j=1, \ldots, n$, $n>1$. Referring to the previous example, two other ways to represent the same model are the following ones:

$$\begin{cases} \text{sum} = \sum_{i=1}^{n} x_i \\ \text{mean} = \dfrac{\text{sum}}{n} \end{cases} \qquad [2] \qquad\qquad \text{result} = \frac{\sum_{k=1}^{z} y_k}{z} \qquad [3]$$

It is intuitive that all previous representations are equivalent, in so far as they "do the same thing". Nevertheless, for our purposes we need a more rigorous definition of equivalence based on the concept of "transformation rule".

We can think to a transformation rule as to a function or procedure whose input is the whole model representation or a part of it, and whose output is a new model representation or a part of it. Obviously, the output of a transformation rule must be semantically consistent with its input. Let us give its formal definition:

Definition 3

> Consider a formal language L and two distinct sets E_1 and E_2 of expressions of L semantically identical. Let R be the set of all transformation rules defined on L; $r \in R$ is defined to be a **transformation rule** on L if applied to E_1 transforms it into E_2.

The existence of transformation rules is very important to state formally the equivalence of model representations. Two equivalent representations must be semantically identical: in other words, there exist two (sets of) transformation rules that transform one into the other, and viceversa. We can formalize the equivalence between model representations as follows:

Definition 4

> Let $S_L = \{ L_j(M_i): j=1, \ldots, n; n>1\}$ be the set of all possible representation of M_i in the language L. Two representations $L_j(M_i)$, $L_k(M_i) \in S_L$, $j \neq k$, are defined to be **equivalent** if there are two sets of transformation rules, R_1 and R_2, defined on L such that R_1 applied to $L_j(M_i)$ transform it into $L_k(M_i)$, and R_2 applied to $L_k(M_i)$ transform it into $L_j(M_i)$. If $R_1 = R_2$ then the two representations are defined **identical**. Obviously, identical representations are also equivalent.

As an example, let us consider two transformation rules, called ***split*** and ***join*** suitable to be applied to representations [1] and [2]. The terms LHS and RHS stand for respectively "left hand side" and "right hand side".

```
transformation split
input
    in_fraction type fraction
output
    out_assignment type assignment statement
    out_fraction type fraction
begin
set RHS of in_assignment to numerator of in_fraction
set numerator of out_fraction to LHS of out_assignment
set denominator of out_fraction to denominator of in_fraction
end
```

```
transformation join
input
    in_assignment type assignment statement
    in_fraction type fraction
output
    out_fraction type fraction
begin
if LHS of in_assignment = numerator of in_fraction then
    exit join
set numerator of out_fraction to RHS of in_assignment
set denominator of out_fraction to denominator of in_fraction
end
```

The rule *split* performs the following operations: given a fraction, it reads its numerator and assigns it to an intermediate variable, and then it builds another fraction whose numerator and denominator are respectively the intermediate variable and the denominator of the given fraction.

The rule *join* acts as follows: given an assignment statement and a fraction whose numerator is the variable on the left hand side of the assignment statement, it builds a new fraction whose numerator and denominator are respectively the right hand side of the assignment statement, and the denominator of the of the given fraction.

Since we can transform representation [1] into representation [2] and vice versa by applying respectively transformation rules *split* and *join*, they are equivalent in the sense expressed in Definition 3. They are not identical, since transformation rules we need to apply are different.

Let us now consider a third rule, called ***rename***, which renames all the elements of a model definition, or a part of them, subject to the simple constraint that all elements with identical name in the input model representation must have identical name in the output one. Model representation [3] is one of the possible results of applying rule *rename* to [1]. Since transformation rule we need to apply to transform representation [1] into representation [3] and vice versa is the same, they are identical.

3. Model Recognition Problem: Basic Ideas.

As asserted in first section of this paper, our main task is to determine which conditions have to be satisfied so that the recognition of a model can be performed. For this purpose, we state that the language L must allow that the set of model representations it produces can be ordered by rank. The rank is a measure, defined on some measurable aspect of L, which allows to class and order model representations. We formalize that as follows:

Definition 5

A formal language L satisfies the property of **rankability** if.

— all model representations $L_j(M_i) \in S_L$ are equivalent;

— all model representations $L_j(M_i) \in S_L$ can be ranked

— S_L can be partitioned by rank and all the elements in the same cell of the partition are identical.

In previous examples we might consider the number of equations as rank. If so, then representation [1] and [3] are of rank 1 while representation [2] is of rank 2. Since all representations are equivalent, and representations [1] and [3] are identical, then the property of rankability holds.

Now, let us explain how recognition process can be carried on. To recognize a model representation means that we have to determine the model it represents. The basic idea of this process is to transform the representation to recognize into another one that we know the kind of model it represents. So doing, we have "recognized" the model.

If the representation we deal with are expressed in a language L satisfying rankability, then all representations of the same model are equivalent. So, if we know all transformation rules that language L allows, then we can recognize any representation simply transforming it into the known one by applying to it the appropriate transformation rules.

The set of all transformation rules may be incredibly large or even not finite. This fact can influence the efficiency of the recognition process. The recognition process can be carried out more efficiently if it is based on the ideas of "minimal representation" and of "reduction rule" defined as follow:

Definition 6

A model representation $L_j(M_i)$ is defined to be **minimal** if:

- L satisfies the property of rankability;
- it has the lowest possible rank.

Definition 7

Given a language L satisfying rankability, **reduction rules** are defined to be transformation rules which when applied to a model representation $L_k(M_i) \in S_L$ of rank *k* produce a model representation $L_j(M_i) \in S_L$ of rank $j < k$.

Referring to previous example, we can consider representations [1] and [2] as minimal ones.

Property of rankability plays a crucial role for our purposes: in fact, if L satisfies rankability, all reduction rules are known, and they form a finite set then:

- it always admit a minimal representation (i.e. a representation which has the lowest possible rank, and to which any other representation of the same model can be reduced);
- any model representation in language L can be reduced in its minimal form (by applying to it the appropriate reduction rule until no more rule can be applied);
- all minimal representations of the same model are identical.

Under the above mentioned condition, the recognition process of a given model representation can be based on the minimal representation by performing the following basic steps:

1) reduce the model representation to recognize to its minimal form;
2) search among the "known" minimal model representation for a template matching the minimal representation obtained by step 1.

Since for any given language L, the set of the reduction rules must necessarily be a subset of the set of the transformation rules, the recognition process of a given model based on the minimal representation is more efficient than the previous one.

Now, we can define formally the condition under which a given model definition language generates "recognizable" model representations:

Definition 8

A model representation is defined to be **recognizable** if the recognition process:
— can be based on its minimal representation
— can be performed in a finite number of steps.

Claim 1

A formal model definition language L generates «recognizable» model representations if:
— it satisfies the property of rankability,
— the set of all reduction rules it admits is finite.

Proof:

If L satisfies property of rankability then it always admit a minimal representation. If the set of the reduction rules is finite any model representation can be reduced to its minimal form in a finite number of steps. In this way both the conditions which state the recognizability of a model representation are satisfied. ■

3. Conclusions

Here we have sketched the fundamental lines to "recognize" models representation. It seems to us that the idea of minimality looks very promising to be further investigated.

Main References

A. ANDRONICO, C. SPERA, P. LOMAGISTRO, *Toward a Structured Modeling System»*, Quaderno del Dipartimento di Matematica, n. 200, August 1991. Submitted to *Decision Support System.*

A.M. GEOFFRION, "An introduction to Structured Modeling", *Management Science*, vol. 33, pp. 547-589.

A CONSTRAINT SATISFACTION APPROACH TO JOB SCHEDULING WITH TIME INTERVAL AND PRECEDENCE RELATIONSHIPS

Anna Sciomachen

Dipartimento Science dell'Informazione
Universita' degli Studi di Milano
Via Comelico 39/41 - 20135 Milano - ITALY

Extended Abstract

This paper describes a simple methodology for reasoning about temporal and precedence constraint satisfiability problems arising in job scheduling. In particular, a Constraint Satisfaction Problem (CSP) approach is presented.

Several researchers, coming both from Artificial Intelligence (AI) and Operations Research (OR) have investigated methods for dealing efficiently with time (see, e.g., [2, 3, 7, 12]); however, at least to the author's knowledge, only very few real and large scale scheduling applications have been approached using this relatively new technique [4].

In this paper, among all the job scheduling problems, an application in which a set V of n jobs has to be processed on a single machine is considered, such that a release date r_i, a deadline d_i and a process time p_i are associated with each job $i \in V$. The problem is formulated on a constraint network, i.e., a digraph $G = (V,A)$ of n nodes (jobs). An arc $(i,j) \in A$ means that job j can be processed immediately after job i. A weight p_j and the attributes r_j and d_j for each node $j \in V$ are given. Moreover, a digraph $P = (V,E)$, with $E \subseteq A$, is given such that an arc $(i,j) \in E$ represents a precedence constraint between

jobs i and j. The problem consists of determining the starting time for processing the jobs in V such that the time windows (defined by r_j and d_j) for scheduling the execution of each node (job) is satisfied and the precedence constraints between nodes given by the relationships defined in arc set E are satisfied within a time horizon (production plan).

Based on the Allen's model for temporal logic [1], a CSP formulation is first presented. A CSP consists of a set of variables $X = \{x_1, x_2, ..., x_n\}$, their associated domains $D_1, D_2, ..., D_n$ and a set C of constraints on these variables. A solution to a CSP consists of an instantiation of all the variables which does not violate any of the constraints. In the case of the application considered in this paper, let X be the set of variables such that x_i represents the starting time for processing job i, $\forall\, i \in V$. A domain D_i is associated with each variable x_i such that $D_i = \{$ set of available Time Machine Units (TMUs) for processing job i (production plan) $\}$. The set C of constraints is defined by two classes of constraints, namely C_1 and C_2, such that $C = C_1 \cup C_2$, $C_1 = \{$ unary constraints (time interval) $\} = \{ r_i \forall i \in V \} \cup \{ d_i \forall i \in V \}$ and $C_2 = \{$ binary constraints (precedences) $\} = \{ (i,j) \in E \}$. The problem is to verify whether an instantiation of all the variables is possible such that all the jobs are completed within their time interval and no precedence relationship is violated.

Starting from the Allen's interval algebra, the temporal relations are specified by atomic relations. In particular, for each pair i,j of jobs the following atomic relations are defined:

- *After(j,i)*: this specifies the precedence relationship between i and j, i.e., $(i,j) \in E$;
- *Available(i,r_i,D_i):* this specifies the release date of job i within the production plan;

- *Due*(i,d_i,D_i): this specifies the deadline of job i within the production plan.

The constraint network G of this problem is then "preprocessed" such that to compute the tightest possible bound for both unary and binary constraints on the jobs. In particular, given the explicit precedence relationships between jobs the possibility of inferring additional implicit precedence relationships are explored; for instance, the transitivity of the predicate After(j,i) may allow to infer information such that

- $After(j,k) \cap After(k,i) \rightarrow After(j,i)$.

Moreover, the availability interval of each job within the production plan is computed by considering its release date, deadline and precedence relationships. The new domain D_i' for each job i in V is hence computed such that the predicate

- $D_i' = Interval(i,r_i,d_i) = D_i \cap Available(i,r_i,D_i) \cap Due(i,d_i,D_i)$

returns the restricted time interval in which each job has to be processed in order to obtain a feasible scheduling of the jobs. Note that all the possible instantiations of the corresponding variables are thus noticeably reduced after the computation of D_i', $\forall$ i $\in$ V.

It is worth mentioning that such a preprocessing approach allows for further generalization of the proposed scheduling problem; for instance, it could be necessary to take into account a possible decomposition of the jobs into different subtasks [13], to analyze periodic scheduling problems [9] or to consider setup times between jobs [5].

A Prolog-like algorithm is then presented for finding a consistent assignment for the variables, i.e., an instantiation of all the variables which does not violate any of the constraints given by both C_1 and C_2. In particular, the procedure

- $x_i = \mathit{Assign}(i, Di)$

associates a value in the new domain D_i' with the corresponding variable x_i, such that a feasible starting time for processing job i is given.

In this phase, following the most-constrained approach suggested in [12], the job having the tightest constraints is selected first. In particular, the procedure

- *Preorder(X)*

performs a sort of the set of variables in such a way that the most critical job, i.e., the most constrained job, is chosen first for its instantiation.

In this particular application the most constrained path is proven to be the most efficient implementative approach, in the sense that the number of backtrackings is minimized (see, e.g., [6, 7] for an overview of the complexity of this kind of temporal CSP problem).

Note that a different way for finding a feasible instantiation of all the variables is to look for an initial solution, possibly inconsistent, and then incrementally repair constraint violations until a consistent assignment is achieved. Such an approach is proposed in [10] in the case of scheduling problems without precedence and time window constraints.

The application field and computational experiences related to real-life cases are also given in the full paper. Some conclusions along with a comparison with a more traditional mathematical programming approach (see, e.g. [5, 8]) for solving the scheduling problem under consideration are finally derived.

References

[1] J.F. Allen, "Maintaining knowledge about temporal intervals", *Communications of ACM* 26 (1983), pp.832-843.

[2] J. Cohen: Constraint Logic Programming Languages - "*Communications of the ACM* 33-7 (1990), pp.52-67.

[3] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier: The Constraint Logic Programming Language CHIP - Proc. Int. Conf. on "Fifth Generation Computer Systems", ICOT, pp.693-702, 1988

[4] M. Dincbas, P. Van Hentenryck, H. Simonis - "Solving the car-sequencing problem in constraint logic programming" - European Conference on Artificial Intelligence, Munich, Germany, August 1988

[5] L. F. Escudero, A. Sciomachen: Local search procdures for improving feasible solutions to the sequential ordering problem - Accepted for publication in "Annals of OR" - G. Mitra, I. Maros Eds, 1992.

[6] M.S. Fox, N. Sadeh, "Why is scheduling difficult? A CSP perspective", Proc. of the 9th European Conf. on Artificial Intelligence, August 1990

[7] M.C. Golumbic, R. Shamir: Complexity and Algorithms for reasoning about time: a Graph-theoretic Approach - Rutcor Research Report RRR 22-91, 1991

[8] E. Hadjiconstantinou, G. Mitra: Transformation of Proposal Calculus Statements into Integer and Mixed Integer Programs: an Approach Towards Automatic Reformulation - Brunel University Research Report TR/11/90, 1990

[9] M. Lacoste, P. Baptiste, "A constraint logic programming approach for the n-periodic hoist schedulinh problem" - *Laboratorie d'automatique* de Besancon, URA CNRS 822, Institut de Productique, 1992

[10] S. Minton, M.D. Johnston, A.B. Philips, P. Laird, "Solving Large-scale constraint satisfaction problems using a heuristic repair method", Proc. 8th National Conf. on Artificial Intelligence, Vol. 1, July 1990

[11] B.A. Nadel: Constraint Satisfaction Algorithms - "*Computational Intelligence*", Vol.5, 1989, pp.188-197.

[12] B.A. Nadel, "Some applications of the constraint satisfaction problem", *Computer Science Dept.* Wayne State University, Detroit, 1990

[13] W. Shih, J.W.S. Liu, J. Chung, "Fast algorithms for scheduling imprecise computations with timing constraints", Report N. UIUCDC-R-89-1509, Dept. Computer Science University of Illinois. Urbana, May 1989